Upset Detection for Passenger Airliners using Classification of Anemometric and Inertial Sensor Data

by

Pieter Jacobus Malan

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in Electric and Electronic Engineering in the Faculty of Engineering at Stellenbosch University

Department of Electric and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Mr J.A.A Engelbrecht
Dr H.A. Engelbrecht

December 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Abstract

Due to the increasing number of civilians making use of airliners for business and travel purposes, safe and reliable operation of commercial passenger airliners is essential. Aircraft upsets are particular conditions that may result in fatal accidents. The accurate identification of such upsets, however, can greatly reduce the risk of fatal accidents.

Conventional flight control systems for aircraft are designed to operate within a specified flight envelope, which is defined by an allowable range of air incidence angles (called the angle of attack and side-slip angle), airspeeds, and angular rates over which the aerodynamic forces and moments behave linearly. The flight envelope is also defined by an allowable range of aircraft attitudes (pitch angles and bank angles) and flight trajectories. When the aircraft exceeds its flight envelope, it enters the upset domain, where the aircraft typically stalls and may enter an uncontrolled spin. A need therefore exists for a system that can detect and recognise "out-of-envelope" conditions using on-board sensor measurements.

Pilots are not always aware that the aircraft has entered the upset domain, especially if visual cues are not available, and may respond with incorrect actions. It is however possible for a pilot to recover from an upset event, but the knowledge of the specific upset event is of utmost importance. A need therefore exists to advise the pilot of an occurring upset so that the correct recovery actions can be taken.

This thesis presents the design and verification of an upset detection system for commercial passenger airliners that detects and identifies flight upset conditions using classification techniques operating on sensor data from on-board anemometric and inertial sensors. The study focused on three aerodynamic upsets, namely high angle of attack upset, underspeed upset and a novel predictive form of high angle of attack upset detection named dynamic pitch upset.

The upset detection system used classification algorithms trained on labelled anemometric and/or inertial sensor measurements. A wide variety of classifiers were investigated in order to determine the feasibility of classification-based upset detection.

The three aerodynamic upset detection systems were evaluated across two cases, namely those that utilised sensor data from both the anemometric and inertial sensors, and those using data from only the inertial sensors. The high angle of attack upset detection system provided an accuracy of 98.8% for initial test case and an accuracy of 92.2% for the latter. The underspeed upset detection system provided an accuracy of 99.3% for the first case and 89.4% for the second case. The dynamic pitch upset detection system provided an accuracy of 93.6% for the initial case and 91.4% for the latter.

The high classifier accuracies provided a suitable and reliable means for aircraft upset detection. These accuracies, along with the locations of the false alarms—the false alarms occur near the upset boundary—enable the pilots to detect and recognise instances pertaining to upset.

Uittreksel

Die toenemende aantal individue wat gebruik maak van lugvaart noodsaak onbetwisbare betroubaarheid van kommersiële-passasier-vliegtuie. Noodlottige vlug ongelukke mag die gevolg van 'n buite vlugbestek toestand wees. Die akkurate identifisering van hierdie buite vlugbestek toestande mag 'n noemenswaardige vermindering in noodlottige ongelukke teweegbring.

Konvensionele vlug-beheerstelsels is ontwerp om binne 'n vasgestelde vlugbestek te funksioneer. Hierdie vlugbestek word gedefinieer as die toelaatbare reeks lugvloei-invalshoeke, lugspoed-intervalle en hoek-tempos waarin die aërodinamiese-kragte en -momente lineêr reageer. Die vlugbestek word ook deur die toelaatbare reeks vliegtuig-oriëntasies en vlugtrajekte bepaal. Indien die vliegtuig die vlugbestek verlaat, betree dit 'n buite vlugbestek toestand, waar die vliegtuig mag staak en onbeheerd-tol. 'n Stelsel wat buite vlugbestek toestande kan identifiseer deur gebruik te maak van beskikbare sensore word dus genoodsaak.

Vlieëniers mag soms nie bewus wees van 'n buite vlugbestek toestand nie, en mag gevolglik met die verkeerde dade reageer. Dit is moontlik om vanuit 'n buite vlugbestek toestand te herstel, maar die kennis van die spesifieke buite vlugbestek toestand is krities in dié verband. 'n Stelsel wat die vlieënier inlig van 'n buite vlugbestek toestand sodat die nodige maatreëls gevolg kan word, is dus van uiterste belang.

Die ontwerp en verifikasie van 'n buite vlugbestek opsporingstelsel vir kommersiële vliegtuie deur die klassifisering van beskikbare sensor metings, word in hierdie tesis aangebied. Drie verskillende buite vlugbestek toestande, naamlik hoë-aanvalshoek, laespoed en dinamiese-hellingshoek was ondersoek.

Die klassifikasie-algoritmes wat ondersoek is, is afgerig op gemerkte anemometries- en inersiële-sensor lesings. 'n Wye verskeidenheid van klassifikasie-algoritmes was ondersoek om die vermoë daarvan as buite vlugbestek opsporingstelsel te bepaal.

Twee gevalle per buite vlugbestek toestand was ondersoek, naamlik anemometries- en inerisiële-sensore beskikbaar, en slegs inersiële-sensore beskikbaar. Die hoë-aanvalshoek-opsporingstelsel het akkuraathede van 98.8% en 92.2% vir die twee gevalle onderskeidelik behaal. Die laespoed-opsporingstelsel het akkuraathede 99.3% en 93.6% vir die twee gevalle onderskeidelik behaal. Die dinamiese-hellingshoek-opsporingstelse het akkuraathede van 93.6% en 91.4% vir die twee gevalle onderskeidelik behaal.

Die hoë akkuraatheid van die stelsels het bygedra tot meer betroubare buite vlugbestek opsporing. Hierdie akkuraathede, gepaard met die posisies van die vals-alarms, (vals-alarms het plaasgevind naby die grens van buite vlugbestek), bemagtig vlieëniers om buite vlugbestek toestande beter te identifiseer.

Acknowledgements

I would like to thank the Airbus Group and the South African National Aerospace Centre for funding this research. I would also like to thank Josep Boada-Bauxell along with his colleagues at Airbus Toulouse for their technical guidance and feedback. Much gratitude is also extended to the project supervisors, Mr Japie Engelbrecht and Dr Herman Engelbrecht for their valuable guidance and insight. All my friends at the ESL for providing regular feedback and advice throughout the project.

Contents

List of Figures

List of Tables

Nomenclature

Constants

g = $9.81\,\mathrm{m/s^2}$

Variables

V_{cas}	Calibrated Airspeed	[kn]
M	Mach number	[ ]
Alt	Altitude	[ft]
α	Angle of Attack	[°]
β	Side-slip Angle	[°]
ϕ	Roll Angle	[°]
θ	Pitch Angle	[°]
ψ	Yaw Angle	[°]
γ	Flight path angle	[°]
P	Roll Rate	[°/s]
Q	Pitch Rate	[°/s]
R	Yaw Rate	[°/s]
a_x	Acceleration in X body axis	[$\mathrm{m/s^2}$]
a_y	Acceleration in Y body axis	[$\mathrm{m/s^2}$]
a_z	Acceleration in Z body axis	[$\mathrm{m/s^2}$]
xCG	Center of gravity position	[%]
$\delta_{ail_{Int}}$	Interior aileron deflection	[°]
$\delta_{ail_{Ext}}$	Exterior aileron deflection	[°]
δ_{THP}	Horizontal stabiliser deflection	[°]
δ_{elv}	Elevator deflection	[°]
δ_{rud}	Rudder deflection	[°]
$\delta_{spoiler}$	Spoiler deflection	[°]
F_X	Force in X body axis	[N]
F_Y	Force in Y body axis	[N]
F_Z	Force in Z body axis	[N]
L	Rolling Moment	[Nm]
M	Pitching Moment	[Nm]
N	Yawing Moment	[Nm]

Abbreviations

AoA	Angle of Attack (deg)
API	Application Program Interface
Bag	Bagging Classifier
DT	Decision Tree
ERR	Equal Error Rate
ESRL	Earth System Research Laboratory
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulations
FDI	Fault Detection and Isolation
INS	Inertial Navigation System
knn	k-Nearest Neighbour
LOC	Loss-of-Control
LR	Logistic Regression
LSB	Least Significant Bit
MCS	Multi-Classifier System
NB	Naïve Bayes
NED	North East Down
NOAA	National Oceanic and Atmospheric Administration
RBF	Radial Basis Function
RF	Random Forest
ROC	Receiver Operator Characteristic
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

This chapter provides and introduction and overview of the research performed in this thesis. The background and motivation for research on flight upset detection is provided, and the concept of flight upset is defined. This is followed by a literature survey of previous research on upset detection for aircraft. The research goal is then stated and elaborated into a list of project objectives. The high level project approach is then described and the main contributions of the research are highlighted. The introduction concludes with an outline of the thesis structure and a summary of the material that will be presented in each chapter.

1.1 Background

Due to the increasing number of civilians making use of airliners for business and travel purposes, safe and reliable operation of commercial passenger airliners is essential. Aircraft upsets are particular conditions that may result in fatal accidents. The accurate identification of such upsets, however, can potentially reduce the occurrence of fatal accidents.

Conventional flight control systems for aircraft are designed to operate within a specified flight envelope, which is defined by an allowable range of air incidence angles (called the angle of attack and side-slip angle), airspeeds, and angular rates over which the aerodynamic forces and moments behave linearly. The flight envelope is also defined by an allowable range of aircraft attitudes (pitch angles and bank angles) and flight trajectories. When the aircraft exceeds its flight envelope, it enters the upset domain, where the aircraft typically stalls and may enter an uncontrolled spin. Pilots are not always aware that the aircraft has entered the upset domain, especially if visual cues are not available, and may respond with incorrect actions. A need therefore exists for a system that can detect and recognise "out-of-envelope" conditions using on-board sensor measurements, and advise the pilot so that the correct recovery actions can be taken.

Typical aerodynamic upset warning systems use measurements of the airflow around the aircraft captured by the anemometric sensors. Anemometric sensors measure the airflow around the aircraft airfoil, this is typically done with tubes that measure air-pressure. These measurements include airspeed, angle of attack and side-slip measurements. Traditional warning systems apply thresholds on the anemometric measurements in order to determine the upset. These thresholds may vary depending on a various input variables such as altitude, airspeed, etc. Traditional warning systems, however, are highly dependent on the data from anemometric sensors. If these (anemometric) sensors fail,

the warning system is rendered useless. A redundancy measure is therefore required to enable reliable aerodynamic upset detection.

This project seeks to investigate and implement reliable aerodynamic upset detection using the classification of sensor measurements from both the anemometric sensors and the inertial sensors. Inertial sensors are sensors based on inertia, and are used to measure linear accelerations and angular rates of an object. This thesis focused on three aerodynamic upsets, namely high angle of attack upset, underspeed upset and a predictive form of high angle of attack upset detection named dynamic pitch upset.

1.2 Definitions of Flight Upset

Defining flight upset is an important first step in developing an upset detection system. The aforementioned definition will be used to establish the motivation and the primary focus for this project. This section initially presents a brief study on the leading cause of fatal large transport aircraft accidents; this serves as the motivation for the research presented in this thesis. Thereafter, flight upsets are defined and discussed.

Aircraft loss-of-control (LOC) is currently the leading cause of fatal large transport aircraft accidents[1]. Loss-of-control is defined as the inability of a pilot to control the aircraft. An investigation to better understand LOC accidents was conducted by Willborn and Foster[2]. Five two-dimensional parameter envelope spaces were defined that contributed to the LOC accidents. These envelopes relate to the aircraft aerodynamics, flight dynamics, structural integrity and flight control use. The five envelopes identified are shown in Figure 1.1 and are listed below:

- *Adverse Aerodynamic Envelope:* angle of attack vs. side-slip angle
- *Unusual Attitude Envelope:* bank angle vs. pitch angle
- *Structural Integrity Envelope:* normal load factor vs. normalised airspeed
- *Dynamic Pitch control Envelope:* dynamic pitch attitude ($\theta + \dot{\theta}\Delta t$) vs. % pitching control command
- *Dynamic Roll control Envelope:* dynamic roll attitude ($\phi + \dot{\phi}\Delta t$) vs. % lateral control command

The adverse aerodynamic envelope defines the limits on the aircraft's angle of attack and side-slip angle. Exiting this envelope typically results in some form of aerodynamic upset.

The unusual attitude envelope is defined by the limits on the bank angle and pitch angle, these limits are based on the generally accepted industry definitions of "unusual attitude". Bank angles greater than $\pm 45°$ and pitch angles higher than 25° nose up and lower than 10° nose down, are considered to be unusual.

The structural integrity envelope is defined by the normal load factor and airspeed limits on an aircraft. The normal load factor limits are prescribed by the Federal Aviation Regulations, based on the structural design requirements of -1 to +2.5 g.

The dynamic pitch control envelope shows the pitch axis control authority (elevator) versus the dynamic pitch attitude. The dynamic pitch attitude is a predicted pitch attitude, based on the current pitch attitude, and its expected change over one second.

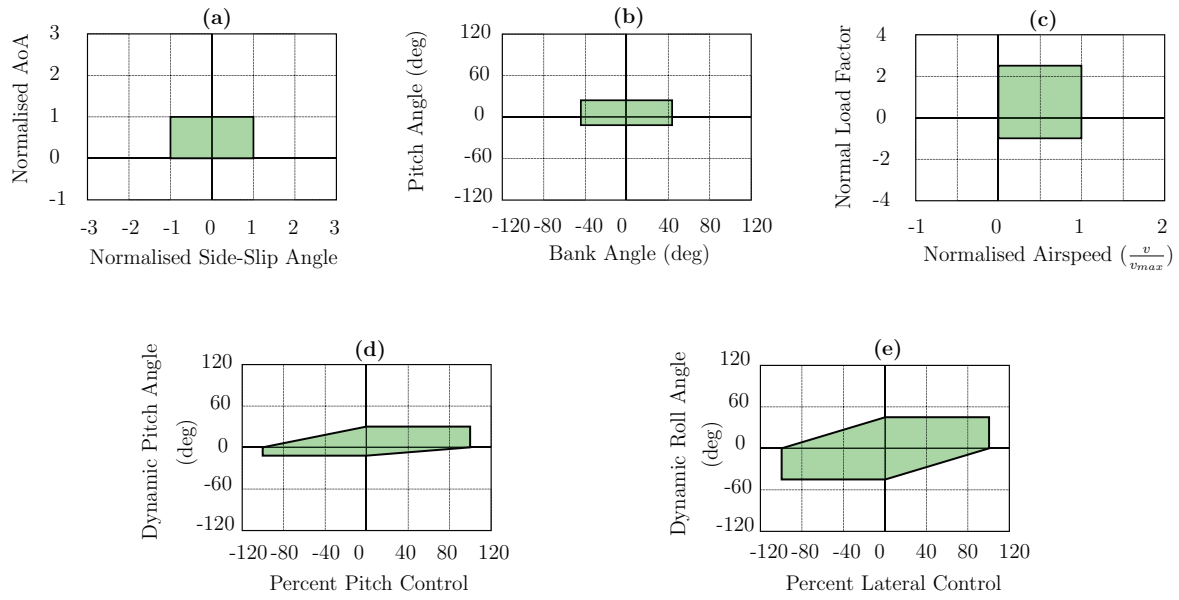

Figure 1.1: The five two-dimensional envelopes: (a)Adverse Aerodynamics Envelope, (b)Unusual Attitude Envelope, (c)Structural Integrity Envelope, (d)Dynamic Pitch Control Envelope, (e)Dynamic Roll Control Envelope, adapted from Willborn and Foster[2].

This envelope indicates whether sufficient actuator authority is available to arrest a pitch motion before the pitch attitude exceeds the unusual attitude envelope.

The dynamic roll control envelope shows the roll axis control authority (aileron) versus the dynamic roll attitude. The dynamic roll attitude is a predicted roll attitude based on the current roll attitude and its expected change over one second. This envelope indicates whether sufficient actuator authority is available to arrest a rolling motion before the roll attitude exceeds the unusual attitude envelope.

Belcastro and Foster continued on this research to identify the causal and contributing factors to LOC accidents. The three main categories identified were adverse on-board conditions, vehicle upsets and external hazards. Vehicle upset contributed to 77.8% of LOC accidents in some way, with the majority of vehicle upsets being aerodynamic upsets [3]. Aircraft upset is defined as the aircraft being outside its normal mode of operation and aerodynamic upset is the aircraft's inability to produce sufficient lift to support its own weight.

Given the aforementioned definition of aerodynamic upset, we can conclude that there is indeed a need for detection and recognition of aerodynamic upset conditions. Such a system will enable the pilot to respond with the appropriate recovery actions.

1.3 Previous Research

The most rudimentary form of upset detection is identification of an aerodynamic upset by a pilot. As an aircraft approaches an aerodynamic upset; the airflow across the upper cambered surface of the wing ceases to flow smoothly and becomes turbulent. The turbulent air then flows across the horizontal stabiliser causing buffeting. This is called pre-stall buffet, and pilots are trained to recognise this effect as a warning sign for an impending upset.

An aerodynamic upset detection/warning system is required on all large transport aircraft, as per section 207 in part 25 of the FAA's Federal Aviation Regulations(FAR) [4]. An audible aerodynamic upset warning system is required in all commercial aircraft; this device sounds an audible warning as a stall speed is approached. Angle of attack sensor measurements are compared with predetermined critical angle of attack angles at different configurations and flight points. A critical angle of attack is the angle of attack where the lift produced by the air-foil reaches a maximum. A stick shaker is a device designed to prevent aerodynamic upset [5]. The stick shaker is activated as the aircraft approaches the critical angle of attack. A stick shaker is a mechanical device that shakes the control column to warn the pilots of an imminent aerodynamic upset.

A form of predictive angle of attack upset detection was proposed by Heinsohn and Neary. The system predicts high angle of attack upset by determining the time rate of change of aircraft's angle of attack. This time rate of change is added to the current angle of attack and compared with a reference level [6].

The majority of the existing research in this research field has been done on Upset Prevention and Recovery Training (UPRT) by the International Committee for Aviation Training in Extended Envelopes (ICATEE). Research done by Advani and Field contained improvements to flight simulators, including an improvement of the surprise factor during an upset [7]. Improvements on the accuracy of the simulation's mathematical model to improve the realism of upsets, and the improvement on motion cueing were the main focus of the research. A combination of an improved flight simulator and an update of the stall training curriculum was proposed to better equip pilots in the event of an upset occurring.

The Simulation of Upset Recovery in Aviation (SUPRA) project, which ties in with the UPRT project, focussed on the improvement of flight simulators used in pilot training [8]. The project included an extended aerodynamic model, capturing the key aerodynamics during and beyond stall for large transport aircraft. New motion cueing solutions for hexapod and centrifuge-based simulation platforms were developed.

The sensors on aircraft have different failure rates due to the functioning of the different sensors. Anemometric sensors have a higher failure rate than inertial sensors [9]. Making use of the more reliable inertial data to estimate anemometric data will add a form of redundancy to an aerodynamic upset detection system, and consequently improve the overall reliability of the system. Estimating anemometric data from the inertial sensors is a popular field of research. The main motivation for the research is to reduce the weight and cost of unmanned aerial vehicles (UAVs) [10]. These estimates of the anemometric data can be used as "virtual sensors" in combination with conventional aerodynamic upset warning systems [11]. The accuracy and precision of the estimate will have an impact on the performance of the warning system. Special consideration should be given to the computational complexity of the estimator, since the estimation algorithm will be executed on the aircraft's flight computer and the flight computer has limited processing power and memory capacity.

The estimation of anemometric sensor data can be split into two main categories. In the fist category, the total body accelerations and the angular rates of the aircraft, along with an aerodynamic model of the aircraft, are used to estimate the anemometric states of the aircraft. Knowing all the forces, moments and the aerodynamic model of the aircraft enables the estimation of anemometric states [12]. In the second category, the anemometric data is calculated from estimates of the ground speed vector and the wind vector.

Oosterom and Babuška proposed a virtual angle of attack sensor in small commercial

aircraft, using the combination of a Takagi-Sugeno fuzzy model and a neural network model [11]. The fuzzy model uses Mach number, dynamic pressure, bank angle, position of the centre of gravity along the X-axis and the aircraft's weight to estimate the trim angle of attack. The neural network is used to estimate the part of the angle of attack signal that can not be modelled by the fuzzy model, which accounts for the high frequency transient angle of attack components. The neural network takes pitch angle, normal acceleration, pitch rate and bank angle as inputs. The proposed system had a root-mean-square error (RMSE) of 0.1125° on the angle of attack estimation. It was, however, not tested for angle of attack angles in the upset domain, where a degradation of accuracy is expected, as the aircraft model is more non-linear at high angle of attack angles. No information regarding the computational complexity was given.

Morelli proposed the real time estimation of aerodynamic parameters without anemometric sensor measurements [12]. The time rate of change of the aircraft velocity vector was determined from the aircraft's current velocity, angular rates, attitude, the total body accelerations, the thrust applied and the aircraft's aerodynamic coefficients [13]. This time rate of change was used to propagate the aircraft velocity that was used to calculate the anemometric data. The system was tested on simulated F16 fighter jet data and on flight data from a sub-scale jet transport aircraft. The system showed promising results, but quantitative estimation error results were not given in the paper.

Ramprasadh and Arya considered the estimation of aerodynamic angles of a mini aerial vehicle in turbulent atmosphere [14]. Their estimation algorithm involved two Extended Kalman filters in a cascade configuration. The first one was used to estimate the aircraft's attitude and the second was used to estimate the angle of attack and the side-slip angle. The angle of attack and side-slip angle were propagated using the total body accelerations, angular rates, mass and an initial angle of attack and side-slip angle. This system had mean errors of on the estimates of the angle of attack and side-slip angle of 1.5° and 0.03° respectively.

Perry et al. used the ground speed vector and an estimate of the wind vector to estimate the anemometric data [10]. The ground speed is measured using the inertial navigation system. The wind vector was estimated from the ground speed vector, aircraft heading and a pitot tube measurement. The difference of the measured ground speed vector and the estimate of the wind vector is used to calculate the angle of attack and side-slip angle. This method assumes horizontal flight; as a result, the performance of the proposed system therefore degraded drastically with highly dynamic flight.

The accuracy of the estimated anemometric data may vary considerably with the quality of the inertial sensors and the accuracy of the aerodynamic model. The dynamics of the manoeuvres performed by the aircraft and the intensity of the wind disturbances have a great influence on the performance of the estimators. Multiple anemometric data estimators exist in the literature, but a quantitative comparison is not possible due to the difference in aircraft, inertial sensors and the dynamics of the manoeuvres performed.

1.4 Research Goal

The research goal was the design and verification of an upset detection system for commercial passenger airliners that detects and identifies flight upset conditions using classification techniques operating on dissimilar sensor data from on-board anemometric and inertial sensors.

1.5 Project Objectives

The research objectives for this project are listed below:

- Detect and recognise aerodynamic upset conditions in commercial passenger aircraft.
- Use inertial sensor data to add redundancy to the upset detection system in case anemometric sensors are not available.
- Investigate the use of classification-based techniques (supervised machine learning) for upset detection.
- Compare the performance of the classification-based approach to upset detection against the more established estimator-based approaches to upset detection found in literature.
- Detect high angle of attack upset conditions, underspeed upsets and dynamic pitch upsets.
- Evaluate the performance of the upset detection system with a statistical analysis of the correct detections, missed detections and false alarms.
- The upset detection function must not provide false alarms in the normal flight domain of the aircraft. A false alarm could cause the pilot to take an inappropriate action, which could lead to an upset.
- Validate and verify the upset detection system with data generated using a representative simulation model of an Airbus A330 with representative sensor noise and environmental disturbances.

1.6 Project Approach

A classification based upset detection system that can detect multiple aerodynamic upsets and operate without anemometric sensors could not be found in the literature. All the current warning systems installed in aircraft have a strong dependence on the anemometric sensors.

An upset detection system detecting and identifying high angle of attack, underspeed and dynamic pitch upsets using classification techniques was developed in this project. Definitions of the upset classes were formulated to ensure the correct labelling of the training and testing data for the classifiers. These definitions formed the core of the system; it was therefore of utmost importance that the definitions be representative of the upsets. Given that flight dynamics knowledge is necessary to fully understand the dynamics of an aircraft entering a stall, a background study of flight dynamics was conducted and used to specify upset definitions, to generate training and test data sets, and to evaluate the classifier results.

Different classification algorithms were investigated to determine the most suitable classifier for each of the individual upset classes. Suitable classification algorithms are presented and discussed to provide some background information on the principles of operation, training, and implementation of the different classifiers.

Data generation for the training and testing of the classifiers was the next focus of the project. The classifiers need to be trained on equal numbers of upset samples and non-upset samples to ensure an unbiased classifier. Two sensor scenarios were considered, one where both anemometric sensors and inertial sensors were available, and one where only inertial sensors were available. The classifier performance was then evaluated in terms of their accuracy (percentages of correct detections, missed detections and false alarms) as well as in terms of the locations of the misidentified samples relative to the normal flight domain and the upset domain. False alarms in the normal flight domain were deemed unacceptable, since they could cause the pilot to take an inappropriate action, which could lead to an upset.

The classifier accuracies at different anemometric sensor noise levels were compared to the accuracy of a conventional upset warning system that simply thresholds the noisy sensor measurements or noisy estimates of the anemometric states. This provides a rough comparison between upset detection using a classification-based approach and upset detection using an estimation-based approach.

The upset detection system was validated by performing simulated upset manoeuvres with the Airbus A330 simulation model that cause the aircraft to enter and then recover from the different types of upset conditions.

1.7 Contributions

The contributions made by this research are listed below:

- A classification based approach to aircraft upset detection was developed. Previous research on upset detection predominately used an estimator based approach.

- An assortment of classification algorithms were considered for the application of upset detection, and their respective performances were evaluated and compared.

- A novel sampling strategy was proposed to generate the training and testing data sets for the classifiers. The random samples are generated to be a union of a wide uniform sample distribution over the entire aircraft state space, and a tight normal distribution about the critical angle of attack surface. The wide uniform sample distribution enables the classifier to correctly distinguish between non-upset and upset conditions that are far away from the critical angle of attack surface, while the normal distribution enables the classifier to accurately discriminate between non-upset and upset conditions for samples that "tightly" surround the critical angle of attack surface.

- A novel form of predictive high angle of attack upset detection was defined and developed.

1.8 Thesis Overview

The overview of the thesis are listed below:

- Chapter 1 gives a detailed description of the problem addressed by this study.
- Chapter 2 presents the necessary background information on flight mechanics and the implementation thereof in a non-linear simulation model.
- Chapter 3 introduces the reader to binary classification and the classification algorithms investigated for upset detection.
- Chapter 4 provides the design and validation of an upset detection system for commercial passenger airliners.
- Chapters 5 to 7 present the design and verification of upset detection functions for high angle of attack upset detection, underspeed detection, and dynamic pitch detection respectively.
- Chapter 8 provides a summary of the research performed and the conclusions reached, and gives recommendations for future research.

Chapter 2

Introduction to Flight Dynamics

The mathematical model and its corresponding simulation model for the A330 aircraft are presented and discussed in this chapter. This project made use of an A330-300 aircraft simulation model received from Airbus. The data generated for classifier training and testing were generated with this model. The labelling models for the three upset classes were also determined form this model. It is therefore important to have a good understanding of flight dynamics and the mathematical model used to simulate the aircraft.

The chapter starts with an introduction to the conventions and notations used in this the simulation model. The six-degree-of-freedom equations of motion with the forces and moments acting on an aircraft are presented to the reader. The Simulink simulation model received from Airbus is lastly discussed.

2.1 Conventions and Notation

Background information such as axis conventions, notations and assumptions are discussed in order to make sense of the aircraft's mathematical model. Figure 2.1 shows the aircraft body axis system, the standard notation and the aerodynamic control surface with the positive deflection angles[13].

2.1.1 Axis Systems

Three axes systems are commonly used when modelling the flight mechanics of aircraft, namely inertial axes, body axes, and wind axes.

2.1.1.1 Inertial Axes

The standard North East Down (NED) axis system (X_E, Y_E, Z_E) is used as the inertial axis system. An inertial reference frame is required to apply Newton's laws of motion which govern the flight mechanics of the aircraft. The NED axes system assumes a flat, non-rotating earth and adequately approximates an inertial reference frame over the short flight distances considered in this project. The origin of the axis is chosen to match a convenient reference point on the earth's surface. The X_E-axis points north, the Y_E-axis is in the east direction and the Z_E-axis points downwards towards the centre of the earth. Gravity is assumed to be in the Z_E direction.

2.1.1.2 Body Axes

The body axis system is fixed to the aircraft body. Figure 2.1 shows the body axis system (X_B, Y_B, Z_B) with its origin chosen to coincide with the aircraft's centre of mass. The body axis system moves and rotates with the aircraft body. The X_B-axis is oriented along the plane of symmetry of the aircraft and is directed towards the front (cockpit) of the aircraft. The Y_B-axis is perpendicular to the plane of symmetry and is directed towards the right-hand wing of the aircraft. The Z_B-axis lies within the plane of symmetry and points downwards relative to the cockpit. Most of the onboard sensor measurements, such as the three-axis gyroscope and accelerometer measurements, are coordinated in the body axis system.

2.1.1.3 Wind Axes

Wind axes are similar to the body axes in that their origin coincides with the centre of mass and therefore move with the aircraft. The orientation of the wind axes differs from the body axes in that the x-axis points in the direction of the velocity vector. The z-axis lies in the aircraft's plane of symmetry and the y-axis is perpendicular to the plane of symmetry and is directed towards the right-hand wing of the aircraft.

2.1.2 Standard Notation

The standard aircraft notation used in this project is shown in Figure 2.1. The forces and translational velocities are shown in red and the moments and angular rates are shown in blue. The notations are as follows,

F_X, F_Y, F_Z:	The coordinates of the force vector in the body axes (longitudinal, lateral and normal force).
L,M,N:	The coordinates of the moment vector in the body axes (rolling, pitching and yawing moment).
U,V,W:	The coordinates of the velocity vector in the body axes (longitudinal, lateral and normal velocity).
P,Q,R:	The coordinates of the angular velocity vector in the body axes (roll, pitch and yaw rate).
$\delta_{ail_{Int}}, \delta_{ail_{Ext}}, \delta_{elv}, \delta_{THP}, \delta_{rud}$:	Inner and outer ailerons, elevator, horizontal stabiliser and rudder control surface deflections respectively. A positive deflection produces a negative moment.

The velocity vector $(\bar{\mathbf{V}})$ of the aircraft may be expressed in spherical coordinates using a velocity magnitude $(\bar{V})$, angle of attack (α) and side-slip angle (β). The relationship between the Cartesian coordinates U, V and W, and the spherical coordinates $\bar{V}$, α and β, of the aircraft velocity vector is given by Equation 2.1.1,

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} \bar{V}\cos\alpha\cos\beta \\ \bar{V}\sin\beta \\ \bar{V}\sin\alpha\cos\beta \end{bmatrix}. \tag{2.1.1}$$

As the speed of the aircraft approaches the speed of sound, the Mach number starts playing an important role in the aerodynamic model. The Mach number is the ratio of

the speed of the aircraft relative to the speed of sound in the surrounding air, expressed mathematically ,

$$M = \frac{\bar{V}}{c(alt)} \ , \tag{2.1.2}$$

where M is the Mach number, $\bar{V}$ is the speed of the aircraft, and $c(alt)$ is the speed of sound at the given altitude.

2.2 Six Degree of Freedom Equations of Motion

A six-degree-of-freedom rigid body model is used to effectively model the motion of an aircraft. The six-degrees-of-freedom equations of motion describe the three-dimensional translation and three-dimensional rotation motion of the body axis system relative to the inertial axis system. A rigid body implies that the position of each mass element on the aircraft remains fixed relative to the body axis system at all times.

The equations of motion can be arranged into two categories, namely kinetics and kinematics. These two categories are discussed in separate sections to provide the reader with necessary information regarding the six-degree-of-freedom model.

2.2.1 Kinetics

Kinetics is the branch of mechanics that relates the forces and moments acting on an object to the kinematic state of the object, with the kinematic states being the position, velocity and acceleration of the aircraft. These relations between forces and moments and the kinematic states are derived from Newton's second law of motion[13]. There are a number of different forms of the kinetic equations of motion depending on which axis systems are used during the coordination of the vectors involved. The equations of motion are presented for the vectors coordinated in the body axes. As the body axis system may rotate with an angular velocity of $\omega_{\mathbf{B}}$ with respect to the inertial space; the Coriolis equation is used to coordinate the inertial rates of change to the body time derivative, as shown in Equation 2.2.1.

$$\frac{d}{dt}(\mathbf{R})]_{Inertial} = \frac{d}{dt}(\mathbf{R})]_{Body} + \omega_{\mathbf{B}} \times \mathbf{R} \tag{2.2.1}$$

where $\mathbf{R}$ is an arbitrary vector.
The kinetic equations were separated into two categories, translational movement and rotational movement, as shown in the remainder of this section.

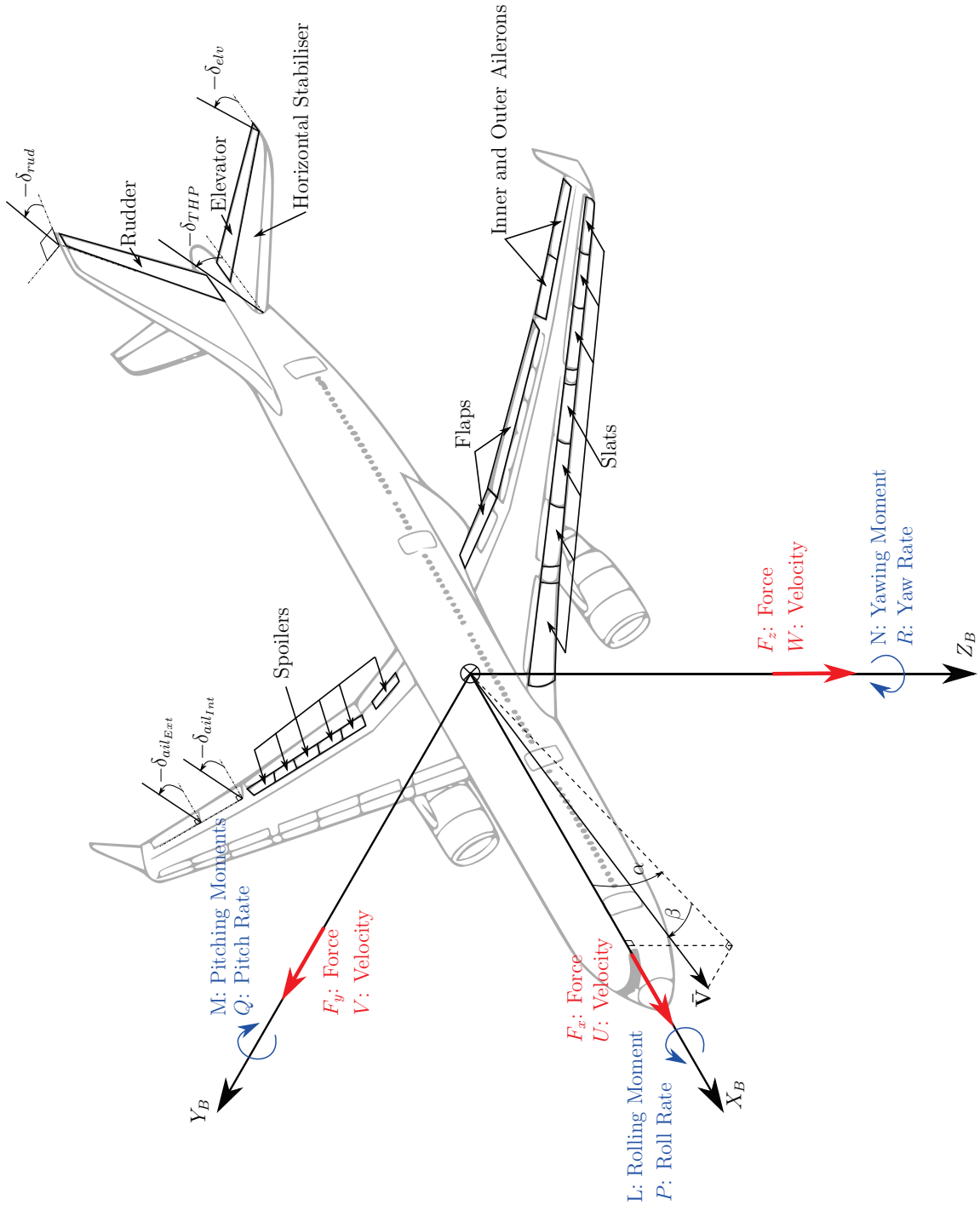

Figure 2.1: Aircraft body axis system and standard aircraft notation with aerodynamic control surface deflections

Translational Motion

Newton's second law of motion states that all the external forces acting on the body must be equal to the rate of change of its momentum as,

$$\mathbf{F_B} = m\mathbf{a_B}]_{Inertial}, \tag{2.2.2}$$

where F_B is the force vector, m is the mass and $\mathbf{a_B}$ is the acceleration vector.

The Coriolis equation shown in Equation 2.2.1 is used to coordinate Equation 2.2.2 to the body axis, as shown in Equation 2.2.3,

$$\mathbf{F_B} = m\mathbf{a_B}]_{Body} + \omega_\mathbf{B} \times (m\mathbf{V_B}), \tag{2.2.3}$$

where $\mathbf{V_B}$ and $\omega_\mathbf{B}$ are the velocity and angular rate vectors of the aircraft.

Expanding Equation 2.2.3 and grouping the forces per axis result in Equation 2.2.4,

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \dot{U} + WQ - VR \\ \dot{V} + UR - WP \\ \dot{W} + VP - UQ \end{bmatrix}. \tag{2.2.4}$$

This shows the relation between the forces that act on an aircraft and the time rate of change of its linear velocity. The various cross terms containing angular rates arise from the coordination of the forces and velocity to the body axes, and not the inertial axis system.

Rewriting Equation 2.2.4 with the rate of change of the velocities, (total body accelerations), as the subject of the equation becomes Equation 2.2.5,

$$\begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix} = \begin{bmatrix} VR - WQ \\ WP - UR \\ UQ - VP \end{bmatrix} + \frac{1}{m} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}. \tag{2.2.5}$$

This is typically used to propagate the velocities of the aircraft over time.

Rotational Motion

Newton's second law for rotational motion states that all the external moments acting on the body must be equal to the rate of change of its angular momentum,

$$\mathbf{M_B} = \mathbf{I_B}\dot{\omega}_\mathbf{B}]_{Inertial}, \tag{2.2.6}$$

with,

$$\mathbf{I_B} = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{xz} & 0 & I_{zz} \end{bmatrix}, \tag{2.2.7}$$

where $\mathbf{I_B}$ is the moment of inertia matrix.

A symmetrical aircraft in the XZ-plane this implies that I_{yz} and I_{xy} are therefore exactly zero, which is an accurate assumption for all conventional aircraft. Applying Equation 2.2.1 to Equation 2.2.6 results in,

$$\mathbf{M_B} = \mathbf{I_B}\dot{\omega}_\mathbf{B}]_{Body} + \omega_\mathbf{B} \times (\mathbf{I_B}\omega_\mathbf{B}), \tag{2.2.8}$$

and, expanding Equation 2.2.8, the three rotational components of the equations of motion are shown in Equation 2.2.9,

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} \dot{P}I_{xx} - \dot{R}I_{xz} + QR(I_{zz} - I_{yy}) - PQI_{xz} \\ \dot{Q}I_{yy} + PR(I_{xx} - I_{zz}) + (P^2 - R^2)I_{xz} \\ \dot{R}I_{zz} - \dot{P}I_{xz} + PQ(I_{yy} - I_{xx}) + QRI_{xz} \end{bmatrix}. \tag{2.2.9}$$

Equation 2.2.10 shows the rate of change of the angular rates. This is used to propagate the angular rates over time,

$$\begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} \frac{I_{xz}(I_{xx}-I_{yy}+I_{zz})PQ+(I_{zz}I_{yy}-I_{zz}^2+I_{xz}^2)QR}{I_{xz}^2-I_{xx}I_{zz}} \\ \frac{(I_{zz}-I_{xx})PR-(P^2-R^2)I_{xz}}{I_{yy}} \\ \frac{I_{xz}(I_{yy}-I_{xx}+I_{zz})QR+(I_{zz}^2-I_{xx}I_{yy}+I_{xz}^2)PQ}{I_{xz}^2-I_{xx}I_{zz}} \end{bmatrix} + \begin{bmatrix} \frac{I_{xz}N-I_{zz}L}{I_{xz}^2-I_{xx}I_{zz}} \\ \frac{M}{I_{yy}} \\ \frac{I_{xz}L-I_{xx}N}{I_{xz}^2-I_{xx}I_{zz}} \end{bmatrix}. \tag{2.2.10}$$

2.2.2 Kinematics

Kinematic equations show the relation among various motion variables, such as linear velocity, angular velocity, attitude and position, to each other over time. An aircraft attitude parametrisation is necessary to represent the body axes with respect to the inertial axis system. The attitude representation is addressed in the following subsection.

Euler 3-2-1 Attitude Representation

A simple and intuitive parametrisation for aircraft attitude is Euler angles. The Euler angle representation is used to describe the attitude of the body axes with respect to the inertial axes, as shown in Figure 2.2. A roll angle (ϕ), is a rotation about the X_B axes, a pitch angle (θ), is a rotation about the Y_B axes and a yaw angle (ψ) is a rotation around the Z_B axes. The Euler 3-2-1 sequence is most commonly used to describe aircraft attitude. This sequence starts with the inertial and body axis aligned and then moves the body axis system though the following set of ordered rotations,

- Yaw the body axis positively through the yaw angle ψ.
- Pitch the body axis positively through the pitch angle θ
- Roll the body axis positively through the roll angle ϕ

The rotations made to represent the body axis system relative to the inertial axis system are shown in Figure 2.3. A transformation matrix using Euler angles may be used to transform the coordinates of a vector in the inertial axis system to the body axis system. This transformation matrix is referred to as the Direction Cosine Matrix (DCM),

$$\begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} = [DCM] \begin{bmatrix} X_I \\ Y_I \\ Z_I \end{bmatrix},$$

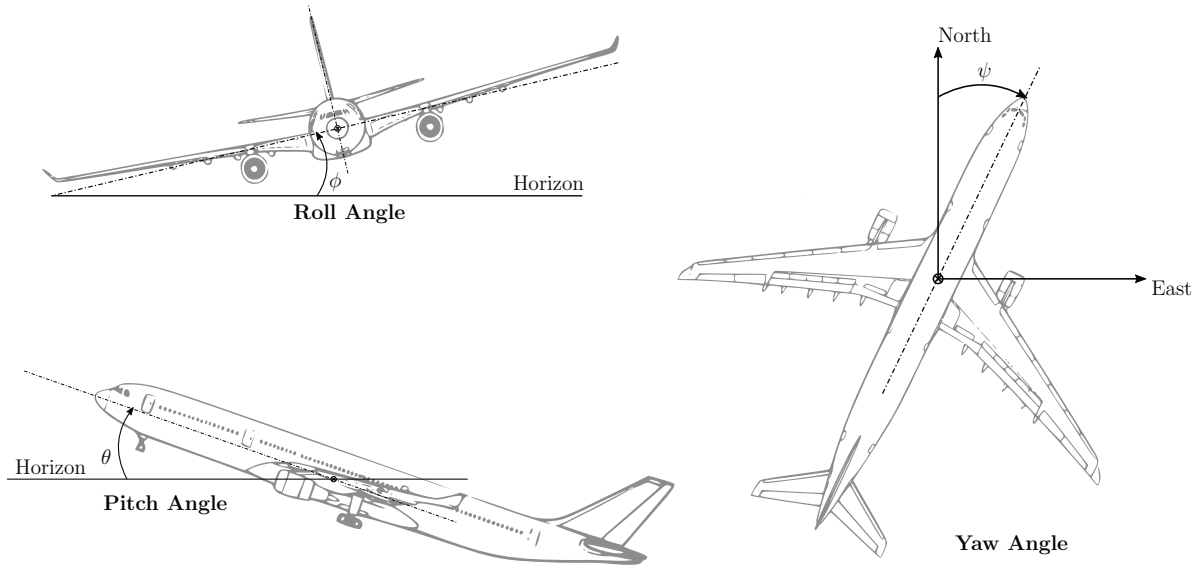

Figure 2.2: Representation of the three Euler angles used to describe the aircraft's attitude

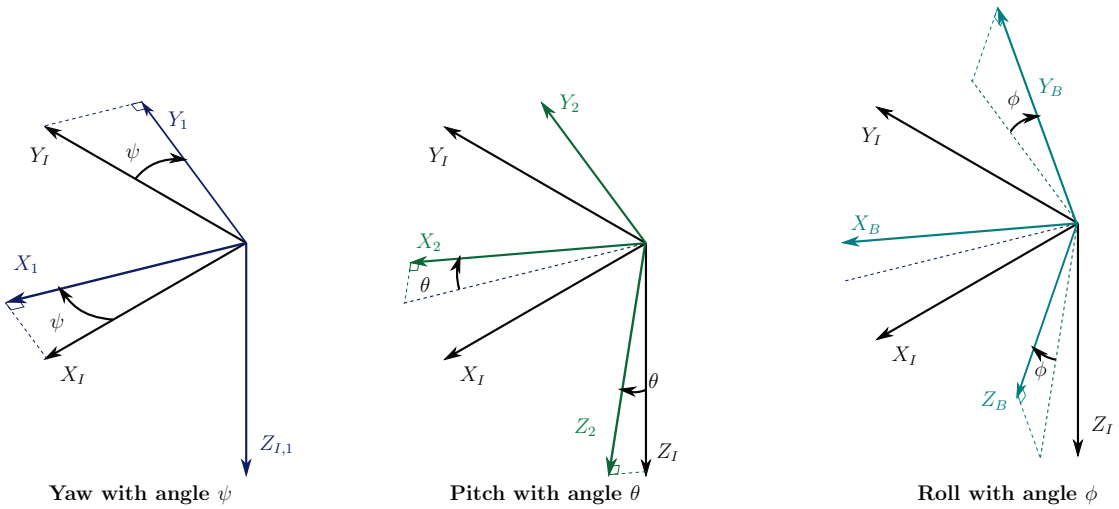

Figure 2.3: Euler 3-2-1 transformation from the inertial axes to the body axes.

with,

$$[DCM] = \begin{bmatrix} C_\psi C_\theta & S_\psi C_\theta & -S_\theta \\ C_\psi S_\theta C_\phi - S_\psi C_\phi & S_\psi S_\theta S_\phi + C_\psi C_\phi & C_\theta S_\phi \\ C_\psi S_\theta C_\phi + S_\psi S_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi & C_\theta C_\phi \end{bmatrix} \tag{2.2.11}$$

where $C_{()} = \cos()$, $S_{()} = \sin()$.

The inverse of the DCM is required in order to transform coordinates from the body axis system to the inertial axis system. Since the DCM is an orthogonal matrix, its matrix inverse is simply its transpose.

Attitude Dynamics

Given this Euler 3-2-1 attitude formulation, it is possible to relate the time rate of change of the aircraft attitude to the body angular rates (PQR) through the following equation,

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}, |\theta| \neq 90°. \tag{2.2.12}$$

It is clear to see from Equation 2.2.12 that for $\theta = 90°$ the $\tan\theta$ and $\sec\theta$ terms become infinite, this is referred to as gimbal lock. Aircraft in upset conditions might experience high roll, pitch and yaw angles and a fully manoeuvrable system is therefore required where there are no restrictions of the pitch angles.

Quaternion Attitude Parametrisation

An alternate method to express an object's orientation is quaternions. Quaternions were first formulated in 1843 by Sir William R. Hamilton and provide an efficient means for updating orientations and more importantly, quaternions do not suffer from a singularity at any attitude. The main disadvantage of quaternions is that they are not as easy and intuitive to understand as the Euler attitude representation. Quaternions can be written as a four-dimensional vector[15],

$$\mathbf{q} = [e_0, e_1, e_2, e_3], \tag{2.2.13}$$

these quaternion parameters are not independent, but are constrained by,

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1. \tag{2.2.14}$$

It can be shown that the DCM shown is Equation 2.2.11, can be written as the following combination of quaternion parameters,

$$[DCM] = \begin{bmatrix} e_0^2 + e_1^2 - e_2^2 - e_3^2 & 2(e_0e_3 + e_1e_2) & 2(e_1e_3 - e_0e_2) \\ 2(e_1e_2 - e_0e_3) & e_0^2 - e_1^2 + e_2^2 - e_3^2 & 2(e_0e_1 + e_2e_3) \\ 2(e_0e_2 + e_1e_3) & 2(e_2e_3 - e_0e_1) & e_0^2 - e_1^2 - e_2^2 + e_3^2 \end{bmatrix} \tag{2.2.15}$$

The transformation from Euler angles to quaternion parameters is as follows,

$$\begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \\ \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \\ \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \\ \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \end{bmatrix} \tag{2.2.16}$$

The time rate of change of the quaternions as a function of the aircraft's angular rates (P, Q, R) are as follows,

$$\begin{bmatrix} \dot{e_0} \\ \dot{e_1} \\ \dot{e_2} \\ \dot{e_3} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -P & -Q & -R \\ P & 0 & R & -Q \\ Q & -R & 0 & P \\ R & Q & -P & 0 \end{bmatrix}\begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix}. \tag{2.2.17}$$

This allows aircraft attitude propagation over time without any restrictions on the attitude of the aircraft. Equations 2.2.15 and 2.2.16 allow conversions between the Euler attitude representation and the quaternion attitude representation.

2.3 Forces and Moments Model

The forces and moments acting on the aircraft must be modelled so that they can serve as inputs to the six-degrees-of-freedom equations of motion discussed in section 2.2. The forces and moments acting on an aircraft can typically be grouped into three categories, namely aerodynamic, propulsion and gravitational. The total resultant force and moment acting on the aircraft is the sum of the forces and moments from each of the three categories. Since the gravitational force acts through the centre of mass of the aircraft, and since the gravity field is assumed to be uniform, the gravitational moment acting on the aircraft is assumed to be zero.

$$\mathbf{F_B} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} F_x^A \\ F_y^A \\ F_z^A \end{bmatrix} + \begin{bmatrix} F_x^T \\ F_y^T \\ F_z^T \end{bmatrix} + \begin{bmatrix} F_x^G \\ F_y^G \\ F_z^G \end{bmatrix}, \tag{2.3.1}$$

$$\mathbf{M_B} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} L^A \\ M^A \\ N^A \end{bmatrix} + \begin{bmatrix} L^T \\ M^T \\ N^T \end{bmatrix}, \tag{2.3.2}$$

where the superscripts A, T and G denote aerodynamic, propulsion and gravitational components respectively. The following subsections will address each of these categories in more detail.

Aerodynamic Forces and Moments

The aerodynamic forces and moments are produced by the relative motion of the air and the aircraft; they are by far the most complex to model and introduce the most uncertainty into the aircraft model. These forces and moments are functions of air-density

(ρ), airspeed ($\bar{V}$), size and geometry of the aircraft, angle of attack (α) and the Mach number (M) at high speeds. The aerodynamic forces and moments are proportional to the dynamic pressure ($\bar{q}$) experienced by the aircraft, as defined below,

$$\bar{q} = \frac{1}{2}\rho\bar{V}^2. \tag{2.3.3}$$

where the air-density, ρ, is dependent on the air-temperature and air-pressure, which are functions of the altitude of the aircraft.

The aerodynamic force and moments are defined below,

$$\begin{bmatrix} F_x^A \\ F_y^A \\ F_z^A \end{bmatrix} = \begin{bmatrix} \bar{q}SC_X \\ \bar{q}SC_Y \\ \bar{q}SC_Z \end{bmatrix}, \tag{2.3.4}$$

$$\begin{bmatrix} L^A \\ M^A \\ N^A \end{bmatrix} = \begin{bmatrix} \bar{q}SbC_l \\ \bar{q}S\bar{c}C_m \\ \bar{q}SbC_n \end{bmatrix}, \tag{2.3.5}$$

where S is the wing reference area, b is the wing span, $\bar{c}$ is the mean aerodynamic chord of the wing and C_X, C_Y, C_Z, C_l, C_m, C_n are non-dimensional aerodynamic force and moment coefficients. The aircraft size parameters are shown in Figure 2.4.

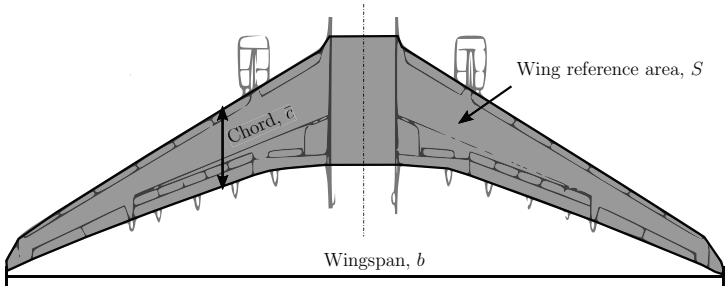

Figure 2.4: The definition of wing reference area S, mean aerodynamic chord $\bar{c}$ and wing span b

The aerodynamic forces and moments are manipulated to control the aircraft. This is done by changing the aerodynamics of the aircraft with actuator deflections ($\delta_{ail}, \delta_{elv}, \delta_{rud}$). The aerodynamics of an aircraft can be modelled as a non-linear function dependent on multiple flight parameters. The components of the aerodynamic forces and moments can be expressed in terms of non-dimensional coefficients, as shown in Equations 2.3.6 and 2.3.7,

$$\begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix} = \begin{bmatrix} C_X(\alpha, \beta, M, Q, Alt, \delta_{THP}, xCG, \delta_{elv}, \delta_{spoiler}) \\ C_Y(\alpha, \beta, M, P, R, Alt, xCG, \delta_{ail_{Int}}, \delta_{ail_{Ext}}, \delta_{rud}, \delta_{spoiler}) \\ C_Z(\alpha, \beta, M, Q, Alt, \delta_{THP}, xCG, \delta_{elv}, \delta_{spoiler}) \end{bmatrix}, \tag{2.3.6}$$

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} = \begin{bmatrix} C_l(\alpha, \beta, M, P, R, Alt, xCG, \delta_{ail_{Int}}, \delta_{ail_{Ext}}, \delta_{rud}, \delta_{spoiler}) \\ C_m(\alpha, \beta, M, Q, Alt, \delta_{THP}, xCG, \delta_{elv}, \delta_{spoiler}) \\ C_n(\alpha, \beta, M, P, R, Alt, xCG, \delta_{ail_{Int}}, \delta_{ail_{Ext}}, \delta_{rud}, \delta_{spoiler}) \end{bmatrix}, \tag{2.3.7}$$

The aerodynamic coefficients used to model the aerodynamics of an Airbus A330 aircraft, are non-linear functions dependent on the following variables,

Linear velocity:

α - Angle of attack (deg),

β - Side-slip angle (deg),

M - Mach number (),

Angular velocity:

P - Roll rate (deg/s),

Q - Pitch rate (deg/s),

R - Yaw rate (deg/s),

Aircraft configuration and altitude:

xCG - Position of the centre of gravity in the X_B axis as a percentage of $\bar{c}$, measured from a reference point (specified by manufacturer) (%),

Alt - Altitude (ft),

Control surface deflections:

$\delta_{ail_{Int}}$ - Interior aileron deflection (deg),

$\delta_{ail_{Ext}}$ - Exterior aileron deflection (deg),

δ_{THP} - Horizontal stabiliser deflection (deg),

δ_{elv} - Elevator deflection (deg),

δ_{rud} - Rudder deflection (deg),

$\delta_{spoiler}$ - Spoiler deflection (deg).

The aerodynamic coefficient functions are implemented as a neural networks that take the above mentioned variables as inputs. These neural networks were trained to fit the aerodynamic model of the A330 based on data generated by simulation, wind-tunnel tests and practical flight tests.

Gravitational Forces and Moments

The gravitational forces and moments acting on the aircraft are modelled using a simple gravitational model that assumes a constant gravitational acceleration that does not vary with latitude or altitude. The standard value for gravitational acceleration is used, and it is assumed to point downwards in the inertial axis system. The gravitational acceleration vector is aligned with the Z_E axis with the origin at the aircraft's centre of gravity, as

shown by Equation 2.3.8,

$$\mathbf{F_E}^G = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}. \tag{2.3.8}$$

Transforming the gravitational force to the body axis system using the DCM in Equation 2.2.11 yields an expression for the gravitational force in body axes shown in Equation 2.3.9. The transformed gravitational force is shown in Figure 2.5.

$$\begin{bmatrix} F_x^G \\ F_y^G \\ F_z^G \end{bmatrix} = \begin{bmatrix} C_\psi C_\theta & S_\psi C_\theta & -S_\theta \\ C_\psi S_\theta C_\phi - S_\psi C_\phi & S_\psi S_\theta S_\phi + C_\psi C_\phi & C_\theta S_\phi \\ C_\psi S_\theta C_\phi + S_\psi S_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = mg \begin{bmatrix} -\sin\theta \\ \cos\theta \sin\phi \\ \cos\theta \cos\phi \end{bmatrix}. \tag{2.3.9}$$

Figure 2.5 shows the decomposition for the forces described in Equation 2.3.9,

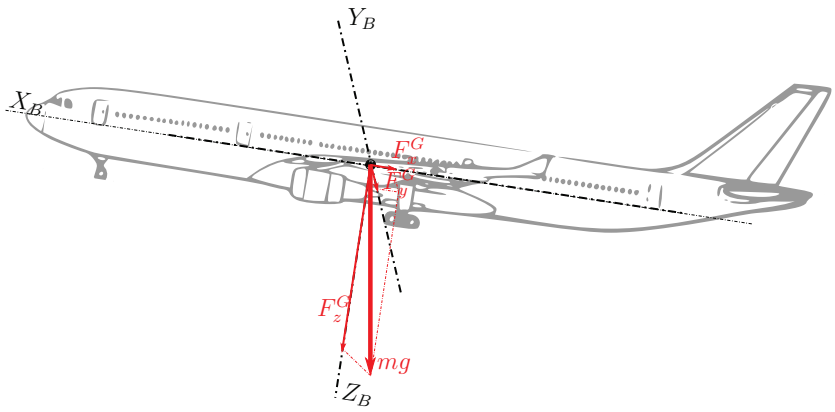

Figure 2.5: The forces in the body axis due to the weight of the aircraft

Since the gravitational force acts through the centre of mass of the aircraft, and since the gravity field is assumed to be uniform, the gravitational moment acting on the aircraft is assumed to be zero,

$$\begin{bmatrix} L^G \\ M^G \\ N^G \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.3.10}$$

Propulsion Forces and Moments

The propulsion of the aircraft is primarily used to gain sufficient airspeed over the aerofoil to produce the lift necessary to overcome gravity. The thrust force produced by the aircraft's engines are aligned with the X_B axis and is displaced with δ_Y^T in the y-direction and δ_Z^T in the z-direction from its centre of gravity. Figure 2.6 shows the thrust force produced and the moments induced due to the engine offsets from the centre of gravity. Equations 2.3.11 and 2.3.12 show the forces and moments generated by the propulsion.

The rolling L^T and yawing moment N^T produced by the engines are zero, due to the symmetrical thrust assumed in this project.

$$\begin{bmatrix} F_x^T \\ F_y^T \\ F_z^T \end{bmatrix} = \begin{bmatrix} T_{left} + T_{right} \\ 0 \\ 0 \end{bmatrix}, \tag{2.3.11}$$

$$\begin{bmatrix} L^T \\ M^T \\ N^T \end{bmatrix} = \begin{bmatrix} 0 \\ (T_{left} + T_{right}) \cdot \delta_Z^T \\ 0 \end{bmatrix}, \tag{2.3.12}$$

where T_{left} and T_{right} are the thrust produced by the left and right engines respectively. A pitching moment is present due to the engine offset in the z-direction.

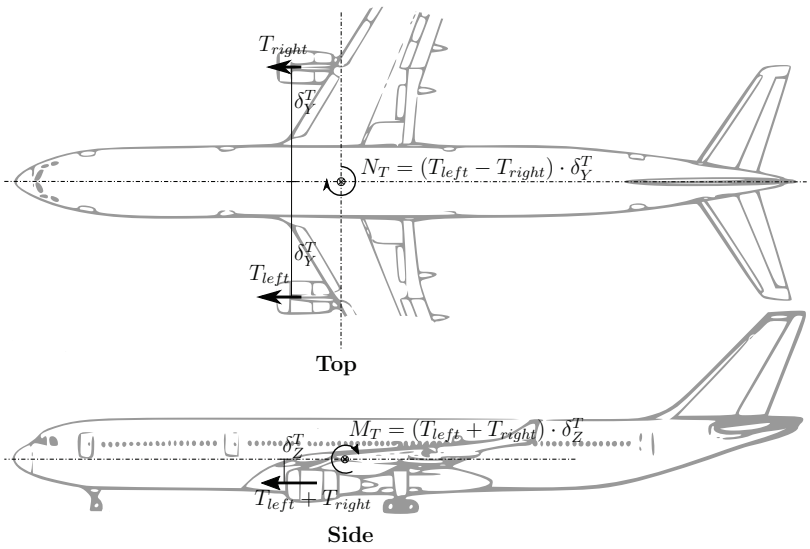

Figure 2.6: The forces and moments produced by the thrust of the aircraft

Mathematical Model Overview

An implementation of the six-degree-of-freedom model for a rigid body with the non-linear forces and moments acting on it, was used in this project. The mathematical model used in this project is shown in Figure 2.7

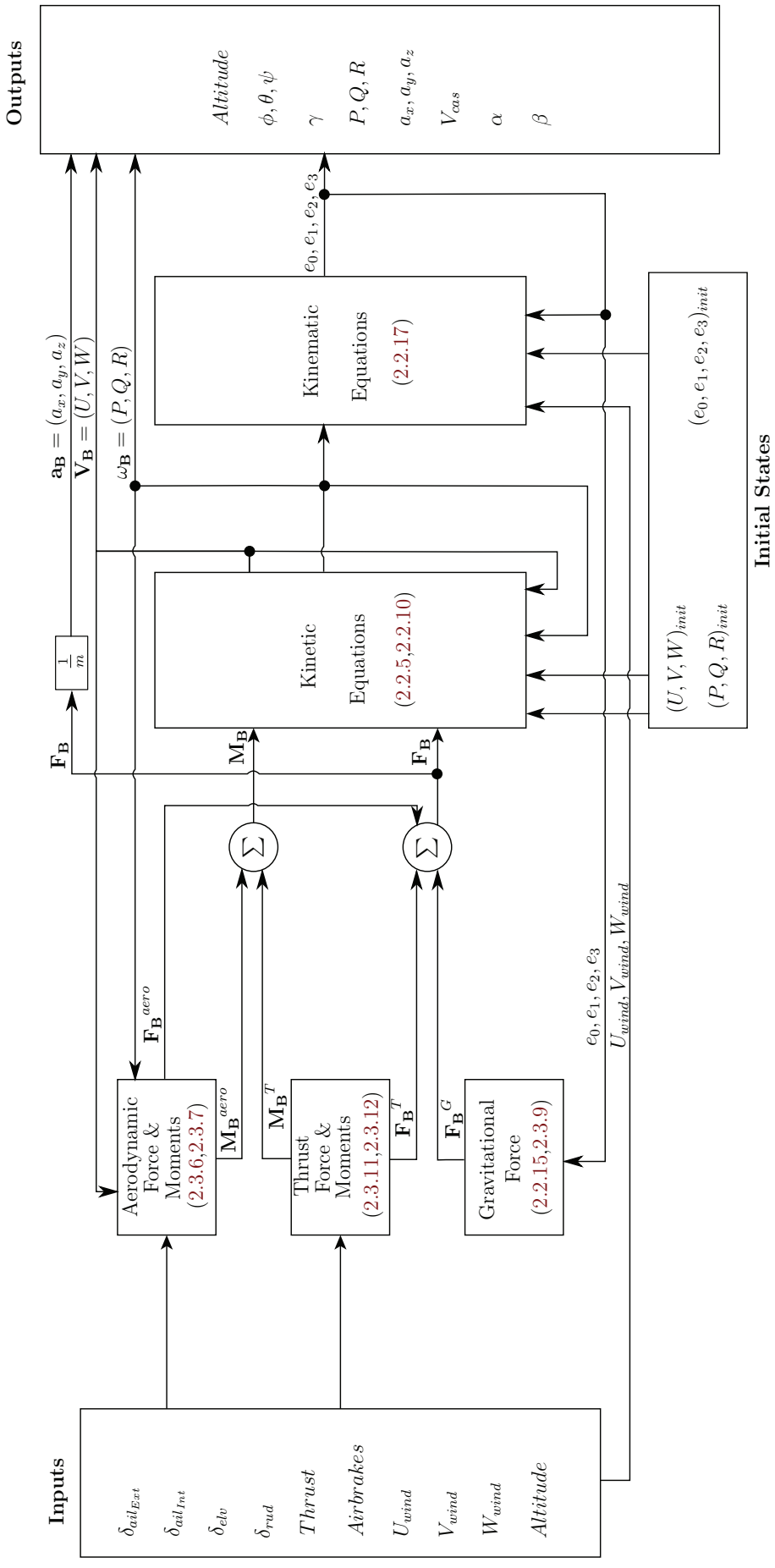

Figure 2.7: Overview of the mathematical model implemented in simulation. [a]

[a]The equations implemented in each step of the simulation are indicated in brackets.

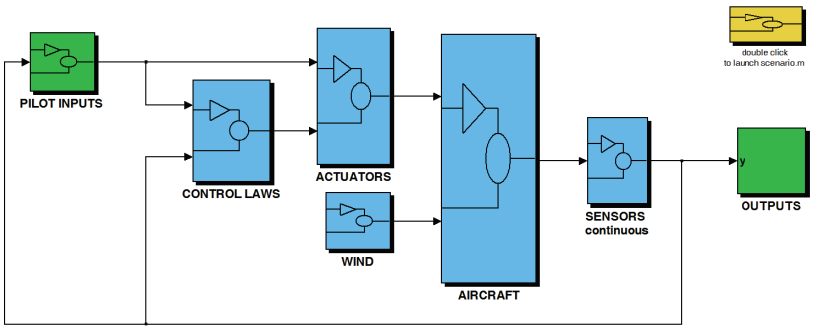

Figure 2.8: Simulink model of an Airbus A330 (top level)

2.4 A330 Simulation Model for Matlab Simulink

Airbus provided a Simulink simulation model that is representative of an Airbus A330 aircraft to the electronic systems laboratory (ESL) for research use. The top level representation of the simulation model consists of seven components, as shown in Figure 2.8. These include pilot inputs, flight control laws, actuator modelling, a non-linear aircraft model, wind disturbances, sensor modelling and lastly the sensor outputs. A brief description of each component block is given to the reader in the following subsections.

Pilot Inputs

The pilot input block consists of two subsections, namely actuator commands and hold modes. A user has the option to give direct actuator deflection reference commands, this simulates the commands received from the control column, pedals and throttle. The actuator inputs to the simulation model are,

δ_{ail} - Aileron deflections,

δ_{elv} - Elevator deflection,

δ_{rud} - Rudder deflection,

Thrust - Thrust supplied by the propulsion system of the aircraft,

Air brakes - Air brake configuration.

The second input option supplied by the simulation model is hold modes. The hold modes are auto-pilot functions that keep the aircraft at specified flight variable values. It calculates the required actuator defection reference to achieve this. The three hold modes supplied are a yaw angle hold mode, flight path angle hold mode and side-slip angle hold mode.

Control Laws

The control laws take the commanded pilot/hold mode references and the aircraft sensor measurements to determined the desired control deflections to achieve these references. The control laws consist of longitudinal and lateral estimators and controllers in order to control the aircraft. These controllers are implemented as S-functions, the inner functions

are therefore concealed to the user. The control law block includes an auto-thrust S-function that is used to generate propulsion commands given a desired airspeed.

Actuators

The actuator dynamics are modelled using a first order low-pass filter, saturation and rate limiting blocks. The low-pass filter is used to introduce the low frequency dynamics introduced by the physical actuators. A saturation block is added to simulate the minimum and maximum actuator deflections. The rate limiter is introduced after the low-pass filter to constrain the rate at which an actuator is allowed to be deflected.

Wind Model

The original simulation model received, allowed the user to specify only a constant wind disturbances. This was found to be inadequate for the purpose of this project. A realistic wind model was therefore implemented. This wind model consisted of three components, the constant wind model, turbulence and wind shear.

The physical science division of the Earth System Research Laboratory (ESRL), which forms part of the U.S. Department of Commerce's National Oceanic and Atmospheric Administration (NOAA), makes available global wind data in the NED axes at different flight levels. Global wind data sets are updated every six hours, at 17 different pressure levels $(1000, 925, 850, 700, 600, 500, 400, 300, 250, 200, 150, 100, 70, 50, 30, 20, 10mb)$ in a $2.5° \times 2.5°$ spatial grid $(0°E \; to \; 357.5°E, \; -90°N \; to \; 90°N)$[16]. The constant wind inputs $(U_{wind}, V_{wind}, W_{wind})$ for the simulation model were based on the global wind figures for 2015. The wind disturbance samples were calculated as uniformly distributed between the 10th and 90th percentile of the wind in each axis. Removing the bottom and top 10th percentiles removes the extreme wind velocities, resulting in a model representative of the typical wind encountered by and passenger airliner. The generated constant wind was orientated in the inertial space was therefore rotated to the body axes.

A von-Kármán turbulence model[17] was implemented to calculate the turbulence associated with the constant wind, velocity vector and altitude. This calculated turbulence was added to the constant wind in order to obtain an accurate representation of the wind disturbance. The wind shear associated with the given wind disturbances were added as disturbances to the angular rates.

Non-linear Aircraft Model

The non-linear aircraft model consisted of the six-degree-of-freedom equations with the aerodynamic, propulsion and gravitational forces and moments, as discussed previously in this chapter. The inputs to the aircraft model are, actuator deflections as output by the control laws, wind disturbances and the aircraft's altitude. The inputs and initial aircraft states (attitude, velocity and angular rates) were used to propagate the aircraft states and to calculate a large variety aircraft sensor measurements as discussed later in this chapter.

Sensor Outputs

The simulation model offers a wide variety of simulation outputs. It was decided to capture the sensor outputs typically available on a large transport aircraft. These outputs

Table 2.2: The sensor measurements captured to be used for upset detection.

Inertial Navigation System	
Altitude	The altitude of the aircraft
ϕ, θ, ψ	The attitude of the aircraft expressed in Euler angles
γ	The aircraft's flight path angle
P, Q, R	The angular rates of the aircraft
a_x, a_y, a_z	The total body acceleration of the aircraft
Anemometric Sensors	
V_{cas}, α, β	The aircraft's relative airspeed vector expressed in polar form, as calibrated airspeed, angle of attack and side-slip angle.

consisted in sensor measurements made available by the inertial navigation system (INS) and the anemometric sensors, as shown in Table 2.2. These sensor measurements were used as the inputs to the upset detection system developed.

Sensor Modelling

The default simulation model received from Airbus did not include any sensor noise, but only modelled the sensor's bandwidth by adding a low-pass filter to the calculated sensor measurement. Adding representative sensor noise to the calculated sensor measurements is an important aspect in ensuring a representative simulation model. Classifier accuracy in the presence of sensor noise is an important performance metric, hence the need for a noise model. It was therefore decided to add noise to the sensor simulated sensor measurements.

Little information was available on the actual noise present on the different aircraft sensor measurements and thus the available sensor information was used to postulate the sensor noise.

The sampling resolution and sampling frequency were known. The sampling resolution was used to calculate the quantisation noise power, as shown in Equation 2.4.1. It was assumed that the quantisation noise forms a small part of the total noise. A second assumption was that the resolution of the analogue to digital converters was chosen large enough to sample the noise on the sensor. An anti-aliasing filter prior to sampling was assumed, resulting in zero mean, band limited Gaussian noise with a frequency band of half of the sampling frequency (f_s)[18].

$$\sigma^2_{quan} = \frac{\Delta^2}{12} \tag{2.4.1}$$

The noise added to the calculated sensor measurements had eight times more power, $\sigma^2_{noise} = 8\sigma^2_{quan}$, than the quantisation noise and was assumed to be zero mean, band limited Gaussian noise with a frequency band of $0.5f_s$. A band limited white noise Simulink block was used to add the noise to the sensors. This block adds a sample from a Gaussian

distribution, as shown in Equation 2.4.2.

$$\begin{aligned} Sensor\ Noise &= \mathcal{N}(\mu = 0, \sigma^2 = (\sigma^2_{noise})(0.5fs)) \\ &= \mathcal{N}(\mu = 0, \sigma^2 = 4\sigma^2_{quan}fs) \end{aligned} \tag{2.4.2}$$

The noise variance of the different sensor measurements captures are shown in Appendix A

Chapter 3

Introduction to Classification

This chapter gives an introduction to classification, this include what classifiers do, how they are trained and what criteria are commonly used to evaluate them. A survey of available classification libraries is conducted and discussed. An overview of classification techniques and a short list of the classifiers that will be used based on the requirements and constraints of the upset detection problem are given. The sort listed classifiers are described in detail. A comparison of the processing and memory requirements of the different classifiers are made.

The information given by this chapter is important to provide the reader with the necessary background on classification. This chapter is used to identify the classification algorithms to be applied to the upset detection problem. It is also used to identify the general approach that will be used to design and verify the classifiers, this include sample generation, labelling, training and testing of the classifiers. This chapter is used to identify the criteria that will be used to evaluate the performance of the classifiers applied to the upset detection problem. And the chapter is lastly used to identify which classifiers will be more feasible to implement within the processing and memory constraints of a flight computer.

The chapter starts with an introduction to classification. The classifier training and typical classification evaluation process is described. A study of the available machine learning libraries is presented. The working of the investigated classification algorithms are presented and discussed. A computational complexity and memory use study for each classifier is presented.

3.1 Classification in Upset Detection

Classification is part of a branch of machine learning called supervised learning. This method is used to determine the category (class) that a new observation (sample) belongs to based on the sample's features[19]. This categorisation is done on the basis of a training data set containing observations of which the class is known.

There exist three main categories of classification algorithms, namely novelty detection classification, binary classification and multi-class classification. These three categories of classification can be seen as classifiers trained to identify one, two or multiple classes respectively. A novelty detection classifier is trained on only one class. Presented with an unknown sample, the classifier calculates the similarity to the known class, and classifies according to this similarity. A binary classifier is trained on data containing two classes. All the samples used to train the classifiers should belong to either of the classes. A

multi-class classifier is trained to identify two or more classes. This is done by training on data containing multiple known classes.

Each classifier category can be divided into two subcategories: non-ensemble and ensemble. Non-ensemble classifiers are algorithms that fit a single classifier or model to the training data. Ensemble classifiers are classifiers that combine multiple non-ensemble classifiers into a single classifier.[20].

The main advantage of non-ensemble classifiers are that they are not as computationally complex as ensemble classifiers. The main disadvantage of non-ensemble classifiers is that they are more prone to over-fitting because they fit the data to a single underlying model and therefore lose generality. Over-fitting is a term used to describe a trained classifier that is fitted too tightly to the training data. This results in poor classifier accuracy, i.e. in classifying observations that are not part of the training data set.[21].

Ensemble classifiers fit multiple classifiers to the training data and combine the classifiers' outputs into a single class output. The main advantage of ensemble classifiers is that they gain generality by fitting the data to multiple underlying models and therefore are less prone to over-fitting than non-ensemble classifiers. The main disadvantage of ensemble classifiers is that their computational complexity increases because they implement and combine multiple classifiers.

Classification as a technique of upset detection was one of the main goals of this project. It was therefore important to investigate as large as possible variety of classification algorithms from one of the classification algorithm categories.

Binary classifiers provide the biggest variety of classification algorithms that rely on a wide variety of different assumptions and, were therefore investigated instead of novelty detection or multi-class classifiers. The focus for the choices of classifiers is to test a wide variety of classifiers that rely on different assumptions. Classifiers of different complexities were investigated. The complexity of the classifiers was taken into consideration, since the upset detection system needs to be implemented on a flight computer. Upset detection can be considered a binary classification problem, as data can be classified into two classes, namely upset and non-upset. For each different type of upset, for example angle of attack upset, underspeed upset, or dynamic pitch upset, a separate binary classifier is trained to detect the specific type of upset. The outputs of each separate upset detection classifier can then be combined to detect and identify different upsets. This study investigated both non-ensemble and ensemble classifiers.

3.2 Classifier Training

A non-linear simulation model, with a complete aerodynamic model of an Airbus A330 as discussed in the previous chapter, are available to generate the training and testing data for the classifiers. There is therefore no limit on the amount of the classifier training data available. The simulation model gives the ability to generate data specifically for binary classifier training for a given upset type.

The classifiers were trained on an equal number of observations from each of two classes, to ensure that the classifiers are unbiased. The training data set must be large enough to ensure that the classifiers are properly trained. A learning curve was used for the purpose of determining the optimal size for the training data set, since a learning curve shows classifier accuracy in relation to the size of the data set. If the training data set is too small, then the classifier will not be trained properly, and will have poor

classification accuracy. If the training data set is too large unnecessary effort is expended in the training process.

3.3 Classifier Evaluation

The main evaluation parameter of classifiers is the percentage of correct classifications, also known as classifier accuracy. There are two types of correct detections, namely true positives and true negatives. True positives are upset samples that are correctly classified as upset and true negatives are non-upset samples that are correctly classified as non-upset. There are two types of incorrect classifications, namely false positives and false negatives. False positives are non-upset samples that are incorrectly classified as upset and therefore represent false alarms. False negatives are upset samples that are incorrectly classified as non-upset samples and therefore represent missed detections. A confusion matrix aggregates all of these variables into one matrix that is used to evaluate a classifier's accuracy. An example of a confusion matrix is shown in Table 3.1 [21].

Table 3.1: Confusion matrix layout

	Predicted Non-Upset	**Predicted Upset**
Actual Non-Upset	True negative	False positive
Actual Upset	False negative	True positive

The majority of classifier outputs are usually a probability value representing a class, i.e. a probability of zero represents non-upset whereas a probability of one represents upset. Classifiers apply a threshold on this probability to classify an observation to a certain class. A classifier can therefore be biased towards a certain class by selecting a threshold that will produce more predictions of one class. Unbiased classifiers were investigated for the purpose of this study, the probability threshold that produce an equal number of false alarms and missed detections was therefore needed. A receiver operator characteristic (ROC) curve is used to determine the percentage of false alarms versus the percentage of missed detections at different thresholds, as shown in Figure 3.1. A ROC curve shows the percentage of false alarms and missed detections at different probability thresholds. The threshold that produces equal percentages of false alarms and missed detections is shown in the figure, the probability threshold for this case was 0.5. An equal error rate (ERR) is when equal percentages of false alarms and missed detections are produced.

A decision boundary is the boundary that separates the two classes, and is created during the classifier training process. These boundaries differ from classifier to classifier and when visualised give great insight into the operation of the classifiers. A decision boundary for a two-dimensional classification problem is used to visualise the differences between the classifiers in this chapter.

3.4 Machine Learning Library Selection

An open source machine learning library was used to identify the top-preforming classifiers. This was decided in order to eliminate obviously poor-performing classifiers, thus

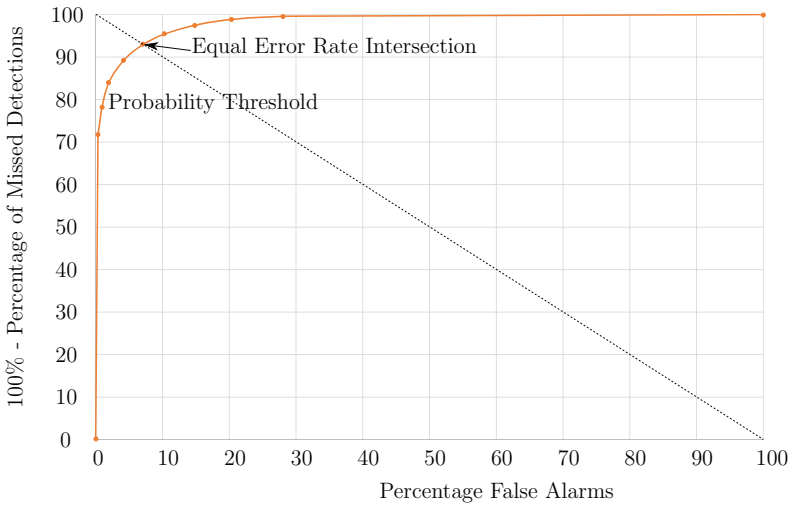

Figure 3.1: An example receiver operator characteristic curve is shown with the equal error rate producing threshold.

saving development time. The top-performing classifiers could then implemented in the programming language of choice.

Many different open-source machine learning libraries are available in many different programming languages. Table 3.2 shows the machine learning libraries considered for this project,

Table 3.2: Machine learning libraries considered in this project

Name	**Supported By**	**Language**
WEKA	University of Waikato	Java
Scikit-learn	Community Project, French Institute of Research in Computer Science and Automation (INRIA) took leadership.	Python
Eureqa Formalize	Nutonian (Not open source, but free student licence)	Proprietary program, able to export mathematical models to any programming language.
MLlib	Apache Spark	Apache Spark pipelines, (supports Java, Python, Scala)
PyLearn2	Community Project, LISA-lab (University of Montreal)	Python
Statistics and Machine Learning Toolbox	MATLAB (Not open source)	MATLAB

The main criteria for selection of the machine learning library were,

- A large variety of classifiers available
- An easy implementation of the available classifiers (a scripting language is preferred)
- A high configurability of the available classifiers
- An open-source library
- A well documented API.

Scikit-learn was used for the implementation of the classifiers. Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms. A machine learning library in Python was chosen due to the benefits of a scripting/interpreted language, helping with the quick and easy implementation of the classification algorithms. Its documented API and the fact that it features numerous classifier training configuration parameters also motivated the selection of this library. It is distributed under the simplified Berkeley Software Distribution (BSD) license [22]. The default classifier configurations were initially used to identify the top-performing classifiers. Scikit-learn offers access to multiple training configuration parameters for the training of the classifiers. A random search within the configuration parameter space was conducted to determine the classifiers' sensitivity to the training parameters, as well as the quality of the default configuration's accuracy. Scikit-learn was used to reduce the time spent on the implementation of classifiers, allowing more time for the generation of the training data and the training of classifiers.

3.5 Non-ensemble Classification

The non-ensemble classifiers used are presented and discussed in this section. The investigated classifiers make use of different fundamental assumptions or classification models that vary in complexity. Since classifier accuracy varies considerably with type of classification problem, a number of different classifiers were tested to ensure the most suitable classifier for upset detection is chosen. The non-ensemble classifiers that were applied to the upset detection problem are shown in Table 3.3,

Table 3.3: Non-ensemble classifiers investigated

Classifier Name	**Classifier Characteristics**
k-Nearest Neighbour (knn)	Instance-based classification
Naïve Bayes Classifier (NB)	Probabilistic classification model
Logistic Regression Classifier (LR)	Generalised linear model
Support Vector Machine (SVM)	Distance-based classification
Decision Tree (DT)	Rule-based classification

The table also shows the different classification models or assumptions used by the classifiers. There is little research available on the use of classification for upset detection, which is why it was decided to investigate a wide variety of classification models.

3.5.1 k-Nearest Neighbour

3.5.1.1 Overview

The nearest neighbour classifier (knn) performs classification based on the shortest distance from the sample to be classified to the known samples on which the classifier was trained. The distances from all the training samples to the unknown sample are calculated and the sample is classified according to the majority vote of a predefined number (k) of shortest distances. Figure 3.3 shows the decision boundary for a two-dimensional classification case.

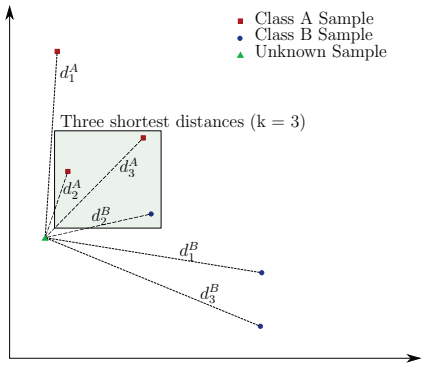

Figure 3.2: The distances between an unknown sample and the training samples. The three shortest distances are shown in the shaded block.

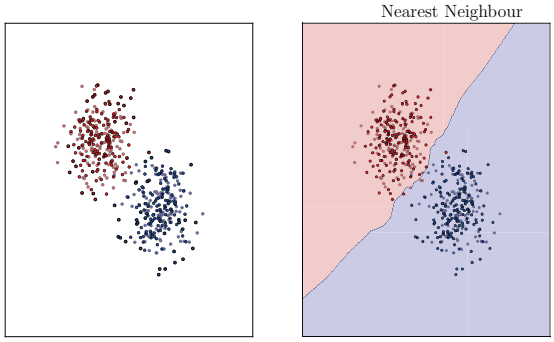

Figure 3.3: Two-dimensional data classified with a k-nearest neighbour classifier. The decision boundary is shown as the blue line in the figure.

3.5.1.2 Training

A k-nearest neighbour classifier is an example of an instance-based learning technique. The classifier uses the actual training data and not a model of the data. There is therefore no classifier training: the storing of the training data in memory can be regarded as the training.

3.5.1.3 Classification

The distances from a training sample to the unknown sample are calculated and used to classify an unknown sample [23]. The sample is classified according to the majority vote of a predefined number (k) of shortest distances.

Figure 3.2 shows the classification process for a knn classifier with k=3 and the Euclidean distance used, with the unknown sample then classified as belonging to class A.

3.5.1.4 Sample Classification Computational Complexity Study

64-bit floating point precision was used to store the training data, this requires eight bytes. The memory required for classification is therefore $8dN$ bytes, with d features and N training samples. It is clear that the memory use of the nearest neighbour classifier grows linearly with the number of training samples. The brute force method of calculating the nearest neighbours is the most computationally complex with, a complexity of $\mathcal{O}(dN)$.

Making use of space partitioning such as the K-D tree can reduce the average complexity to $\mathcal{O}(logN)$.

3.5.1.5 Advantages and Disadvantages

The advantage of the knn classifier is its conceptual and computational simplicity. The main disadvantage is that the knn classifier is memory-intensive when trained on large data sets [24].

3.5.1.6 Scikit-Learn Training Configuration Parameters

The Scikit-learn training configuration parameters are,

- The number of neighbours used in the majority vote
- The nearest neighbour algorithm used
- The distance metric used.

The number of neighbours used by the classifier can be set, with the default used being five neighbours. The number of neighbours will have a smoothing effect on the decision boundary proportional to the number of neighbours. There is an upper bound where classifier accuracy is reduced by the number of neighbours added to the voting proses.

Several algorithms are included in the library to determine the nearest neighbours. The brute force method involves the computation of the distance between all pairs of points in the data set. The brute force method may be a worthwhile option for small data sets, but becomes impractical for large data sets. In order to address the computational inefficiencies of the brute force method, a variety of tree-based structures have been invented. These trees operate on the assumption that if it is known that points X and Y are far from each other, and that points Y and Z are close to each other, then it can be assumed that points X and Z are far from each other. The two tree structures included in the Scikit-learn library are a K-D tree and a Ball tree. The construction of the K-D tree is fast, but becomes inefficient as the number of features grows. Ball trees on the other hand are costly to construct, but can be very efficient on highly-structured data, even for many features. The number of features (sensor measurements) used for upset detection is relatively small, making the K-D tree the preferred option.

Numerous popular distance metrics are available to be used with the knn classification algorithm. The distance metrics available are euclidean, manhattan, chebyshev, minkowski and mahalanobis.

3.5.2 Naïve Bayes

3.5.2.1 Overview

The Naïve Bayes classifier is based on the probabilistic model derived from Bayes' theorem [25]. It calculates the probability that a sample belongs to a class based on each of its feature value probabilities. The Naïve Bayes classifier assumes feature independence when calculating probabilities. This assumption that features are independent is why the classifier is called "naïve" [24]. Some features used for upset detection are not independent of each other, and this might influence the performance of the Naïve Bayes classifier.

Figure 3.4 shows the decision boundary for the two-dimensional data. The classifier assumes a Gaussian distribution, as can be seen in the figure.

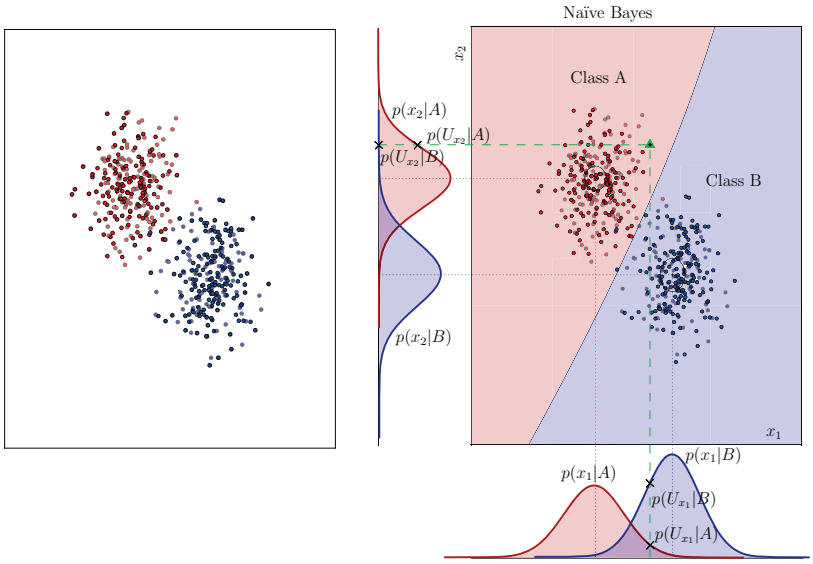

Figure 3.4: Two dimensional data classified with a Naïve Bayes classifier. The decision boundary is shown as the blue line in the figure. The feature distributions are also shown.

3.5.2.2 Training

Axes-independent Gaussian distributions are assumed for the spread of the classes. This implies a diagonal covariance matrix. The classifier training process consists of determining the description of the Gaussian distributions that fits the data belonging to each of the two classes. The variance (σ) and mean (μ) are calculated in each feature dimension of the training data. The training process fits conditional probabilities to the data given the class, as shown by the one-dimensional Gaussian distributions ($p(x_1|A), p(x_1|B), p(x_2|A)$ and $p(x_2|B)$) in Figure 3.4.

3.5.2.3 Classification

The probability of a sample belonging to a class can be calculated using Bayes' theorem and the independence assumption shown in Equation 3.5.1,

$$p(\bar{U}|y) = \prod_{i=1}^{d} p(U_i|y), \tag{3.5.1}$$

with $\bar{U}$ being a d-feature unknown sample. Figure 3.4 shows the conditional probabilities of the two-dimensional unknown sample (U) with the following labels, $p(U_{x_1}|A)$, $p(U_{x_2}|A)$, $p(U_{x_1}|B)$, $p(U_{x_2}|B)$.

The probability distribution of an unknown sample given the class is calculated for both classes using Equation 3.5.1, as shown in Equations 3.5.2 and 3.5.3.

$$p(\bar{U}|A) = p(U_{x_1}|A)p(U_{x_2}|A) \tag{3.5.2}$$

$$p(\bar{U}|B) = p(U_{x_1}|B)p(U_{x_2}|B) \tag{3.5.3}$$

The probability of a class given the unknown sample is calculated using Equations 3.5.2 and 3.5.3 and Bayes' theorem, as shown in Equations 3.5.4 and 3.5.5.

$$p(A|\bar{U}) = \eta p(\bar{U}|A)p(A), \tag{3.5.4}$$

$$p(B|\bar{U}) = \eta p(\bar{U}|B)p(B). \tag{3.5.5}$$

with η being the scaling coefficient.

The sample is classified into one of the two classes based on the conditional probabilities calculated in Equations 3.5.4 and 3.5.5. The unknown sample is classified as belonging to the class for which it has the largest conditional probability. This example can be extended to d features.

3.5.2.4 Sample Classification Computational Complexity Study

The memory footprint of the Naïve Bayes classifier is small compared to the other classifiers. The mean and variance for each dimension are the only parameters that needs to be stored. It can be shown that the memory required by the classifier is $16d$ bytes, if the mean and variance are stored as 64-bit floating point precision, where d is the number of features. The memory required does not increase with an increase in the number of training samples. The computational complexity for the classification of an unknown sample grows linearly with the number features used, $\mathcal{O}(d)$.

3.5.2.5 Advantages and Disadvantages

The advantage of the Naïve Bayes classifier is that training and classification of data are simple to execute with little computational complexity, which makes it ideal to be executed on a platform with little computational resources available. The disadvantage of the Naive Bayes classifier is that it assumes independence between the features used for the classification, which may not be a valid assumption, and this may lead to poor classifier performance.

3.5.2.6 Scikit-Learn Training Configuration Parameters

There are no Scikit-Learn configuration parameters for a Gaussian Naïve Bayes classifier.

3.5.3 Logistic Regression

3.5.3.1 Overview

Logistic regression calculates the relationship between a dependent variable and a number of the independent variables by estimating probabilities using a logistic function. Logistic regression is used as a classifier by applying a threshold to the estimate of the dependent variable given the independent features. The class label of the sample is considered as the dependant variable and the features of the samples are the independent variables. A function with an output that lies between zero and one is needed to represent a binary class. Logistic regression uses the logistic function shown in Equation 3.5.6 to estimate the class label of the unknown samples [21],

$$h_\theta(\bar{x}) = \frac{1}{1 + e^{\bar{\theta}^T \bar{x}}}, \tag{3.5.6}$$

where $\bar{\theta}$ are the regression coefficients that need to be determined to ensure a well-fitted sigmoid function to the data. The independent feature values of the unknown sample are given by the vector, $\bar{x}$.

Figure 3.5 shows the decision boundary of the logistic regression classifier given a two-dimensional classification case.

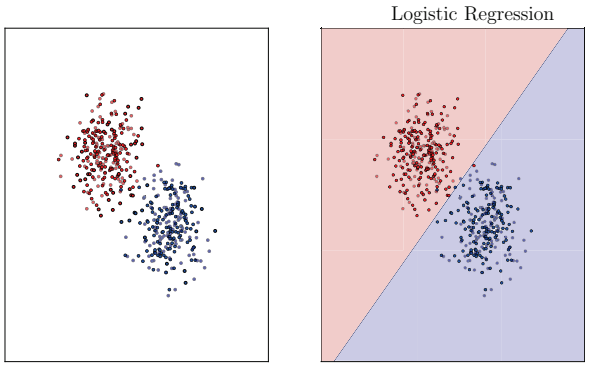

Figure 3.5: Two-dimensional data classified with a logistic regression classifier. The decision boundary is shown as the blue line in the figure.

3.5.3.2 Training

The regression coefficients of the logistic function is calculated in the training process by determining the minimum of a cost function. A convex cost function is necessary to determine the regression coefficients, without the risk of the optimisation algorithm converging at a local minimum. A convex function is a function with an absolute optimum. Equation 3.5.7 shows the convex cost function used, with y being the label of the sample. A gradient descent method is used to determine $\bar{\theta}$ that will result in $J(\theta)$ reaching an absolute minimum, with the dimension of $\bar{\theta}$ being the dimension of the features used. There are multiple gradient descent algorithms to minimise the cost function, J(θ)

$$Cost(h_\theta(\bar{x}), \bar{y}) = -\bar{y}\log(h_\theta(\bar{x})) - (1-\bar{y})\log(1-h_\theta(\bar{x})), \tag{3.5.7}$$

$$J(\theta) = \frac{1}{d}\sum_{i=1}^{d} Cost(h_\theta(x^i), y^i).$$

Regulation of the θ is necessary to prevent over-fitting of the regression model. Scikit-Learn offers L1 and L2 regularisation. L1 and L2 regularisation methods shrink the estimate of θ towards zero relative to the maximum likelihood estimates. The L2 penalty tends to shrink some coefficients to small but non-zero values [26], whereas L1 regularisation shrinks most of the regression coefficients to exactly zero and with relatively little change to the more important regression coefficients[27].

3.5.3.3 Classification

Classification is done by evaluating Equation 3.5.6 at the unknown sample. The output of Equation 3.5.6 lies between 0 and 1, with 0 being non-upset and 1 being upset. A threshold is applied to an output of 0.5, with all the values less than 0.5 being non-upset and all the values greater than and equal to 0.5 being upset.

3.5.3.4 Sample Classification Computational Complexity Study

The memory required by the classifier is determined by the dimension of the regression coefficients, $\bar{\theta}$. The dimension of the regression coefficients is the same as the number of features used, and it can therefore be shown that the memory required by the classifier is $8d$ bytes. The computational complexity of the classifier depends mainly on the number of features used per sample, indicated with $\mathcal{O}(d)$.

3.5.3.5 Advantages and Disadvantages

The advantage of the logistic regression approach is that the classification of an unknown sample requires little computational resources. The disadvantages of the logistic regression approach are that the classifier might over-fit if regulation is not adequately applied to the classifier. The linear model might not hold well for complex non-linear models. The two classes might be linearly inseparable, in which case the logistic regression classifiers will not be able to classify the data.

3.5.3.6 Scikit-Learn Training Configuration Parameters

The Scikit-learn training configuration parameters are,

- The regulation algorithms used on the regression coefficients
- The optimisation algorithm used
- The regulation strength used
- The maximum number of iterations used in the training
- The stopping criteria tolerance used.

L1 or L2 regularisation of the regression coefficients can be selected. The gradient descent algorithms used on the cost function to determine the regression coefficients are newton-cg, the Broyden-Fletcher-Goldfard-Shanno algorithm (lbfgs) and Liblinear. The regulation strength, C, can be changed to reduce over-fitting at the cost of performance. The maximum number of iterations taken for the solvers to converge can be limited to reduce training time. The tolerance for the stopping criteria can be changed.

3.5.4 Support Vector Machine

3.5.4.1 Overview

A sample can be seen as a point in a multidimensional space, with the number of dimensions equal to the number of features of the sample. The model built by a support vector machine (SVM) can be interpreted as a surface (hyperplane) separating the two classes in this multidimensional space. The support vector machine is a learning method that attempts to fit this hyperplane for optimal separation between the two classes. A hyperplane's position is typically determined by the maximum distance from both classes [28].

Figure 3.6 shows two classes separated by a linear hyperplane for a two-dimensional classification problem. The functional margin is the distance to the nearest data points of any class. The hyperplane is chosen to fall in the middle of the largest functional margin.

It turns into an optimisation problem to find the widest functional margin, as shown in Figure 3.6.

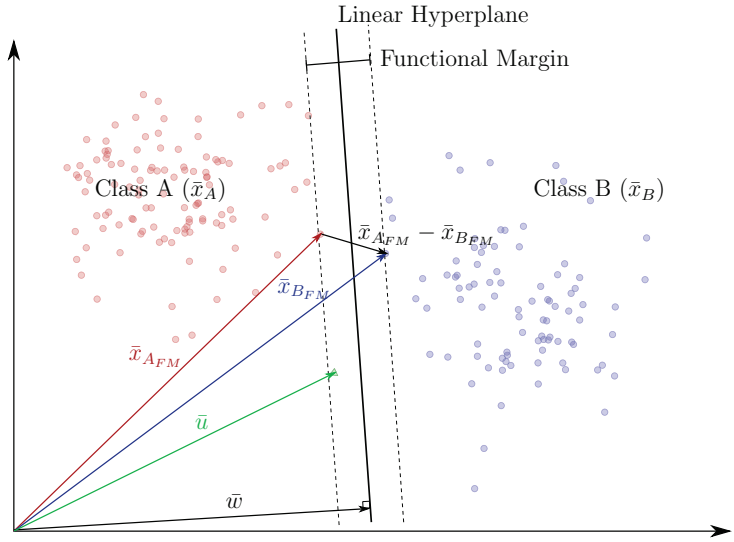

Figure 3.6: Binary class classification using a SVM with a linear hyperplane. The largest functional margin separating the two classes is shown.

3.5.4.2 Training

The support vector machine training process fits a hyperplane to separate the two classes in the multidimensional feature space. The hyperplane is chosen to fall in the middle of the largest functional margin. Lagrange multipliers are used to determine this largest functional margin.

The projection of the known samples, $\bar{x}_B$ and $\bar{x}_A$, on a vector perpendicular to the hyperplane, $\bar{w}$, is:

$$\bar{w} \cdot \bar{x}_B + b \geq 1, \tag{3.5.8}$$

$$\bar{w} \cdot \bar{x}_A + b \leq -1. \tag{3.5.9}$$

where b is an arbitrary threshold.

For the sake of convenience, let us define y_i for the two classes:

$$y_i = \begin{cases} 1, \ for\ Class\ samples\ B \\ -1, for\ Class\ samples\ A \end{cases} \tag{3.5.10}$$

Combining Equations 3.5.8, 3.5.9 and 3.5.10 results in a general expression for the projection of the known samples ($\bar{x}_i$) on $\bar{w}$, as shown in Equation 3.5.11:

$$y_i(\bar{x}_i \cdot \bar{w} + b) \geq 0. \tag{3.5.11}$$

For the samples that lie on the boundary of the functional margin (support vectors), $\bar{x}_{i_{FM}}$, –shown in Figure 3.6– Equation 3.5.11 can be simplified to Equations 3.5.12 or 3.5.13.

$$y_i(\bar{x}_{i_{FM}} \cdot \bar{w} + b) - 1 = 0, \tag{3.5.12}$$

$$\bar{x_{i_{FM}}} \cdot \bar{w} = \begin{cases} 1 - b, & for\ Class\ B \\ 1 + b, & for\ Class\ A \end{cases}. \tag{3.5.13}$$

An expression for the width of the functional margin, FM_{width}, is shown in Equation 3.5.14.

$$FM_{width} = (\bar{x}_{A_{FM}} - \bar{x}_{B_{FM}}) \cdot \frac{\bar{w}}{\|\bar{w}\|}. \tag{3.5.14}$$

Combining Equations 3.5.13 and 3.5.14, the width can be expressed as Equation 3.5.15,

$$FM_{width} = \frac{2}{\|\bar{w}\|}. \tag{3.5.15}$$

To determine the maximum width of the functional margin, it is necessary to determine the maximum of Equation 3.5.15, or similarly determine the minimum of $\|\bar{w}\|$. Determining $\min \frac{1}{2}\|\bar{w}\|^2$ makes it possible to determine the extreme minimum by using Lagrange multipliers shown in Equation 3.5.16, with Equation 3.5.12 as the constraint for the optimisation problem [29].

$$\mathcal{L}(\bar{w}, b) = \frac{1}{2}\|\bar{w}\|^2 - \sum_i \alpha_i[y_i(\bar{w} \cdot \bar{x}_{i_{FM}} + b) - 1] \tag{3.5.16}$$

where $\mathcal{L}()$ is the Lagrange function and α_i is a Lagrange multiplier.

The turning point of Equation 3.5.16 is shown in Equation 3.5.17,

$$\nabla_{(\bar{w},b)}\mathcal{L}(\bar{w}, b) = 0 \Leftrightarrow \begin{cases} \bar{w} - \sum_i \alpha_i y_i \bar{x}_{i_{FM}} = 0 \\ -\sum_i \alpha_i yi = 0 \end{cases}. \tag{3.5.17}$$

It is clear from Equation 3.5.17 that,

$$\bar{w} = \sum_i \alpha_i y_i \bar{x}_{i_{FM}} \tag{3.5.18}$$

Substituting Equation 3.5.18 into Equation 3.5.16 yields,

$$\mathcal{L}(\bar{w}, b) = \frac{1}{2}(\sum_i \alpha_i y_i \bar{x}_{i_{FM}})(\sum_j \alpha_j y_j \bar{x}_{j_{FM}}) - \sum_i \alpha_i y_i \bar{x}_{i_{FM}} \cdot (\sum_j \alpha_j y_j \bar{x}_{j_{FM}}) - \sum_i \alpha_i y_i b + \sum_i \alpha_i$$

, which can be simplified by using Equation 3.5.17, as shown in Equation 3.5.19,

$$\mathcal{L}(\bar{w}, b) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{x}_{i_{FM}} \cdot \bar{x}_{j_{FM}} \tag{3.5.19}$$

The optimisation problem is reduced to finding the maximum of $\bar{x}_{i_{FM}} \cdot \bar{x}_{j_{FM}}$.

In some cases the training samples might be linearly inseparable and the optimisation algorithms will not be able to find a maximum for $\bar{x}_{i_{FM}} \cdot \bar{x}_{j_{FM}}$. The samples can be transformed with a transformation function, $\phi(\bar{x})$, into a different space so that

a optimal hyperplane can be fitted. The optimisation still depends on a dot product, therefore as long as $\phi(\bar{x}_i) \cdot \phi(\bar{x}_j)$ can be determined it is not necessary to determine the actual transformation function. It is thus only necessary to optimise the kernel function, $K(\bar{x}_i, \bar{x}_j) = \phi(\bar{x}_i) \cdot \phi(\bar{x}_j)$, to find the support vectors and classify unknown samples. Applying a kernel results in the shape of the hyperplane changing when transformed back to the original sample space, as shown in Figure 3.7 for a two-dimensional case. Figure 3.7 shows the effect of different kernels on the decision boundary of a SVM classifier [28].

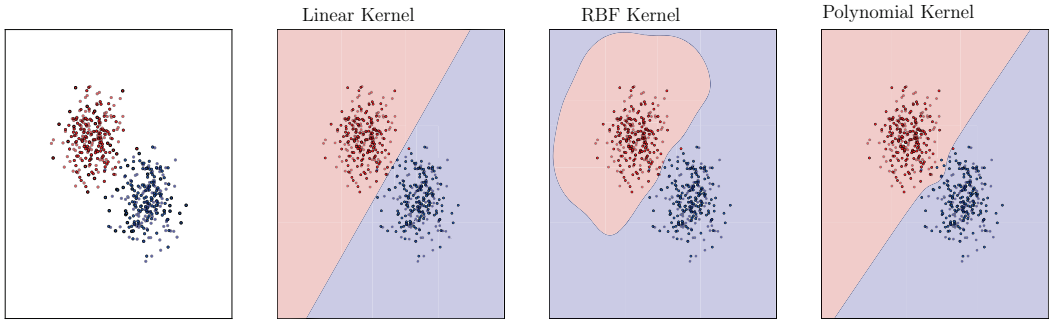

Figure 3.7: Two-dimensional data classified with SVMs using different kernels. The decision boundary is shown as the blue line in the figure.

3.5.4.3 Classification

Let $\bar{w}$ be a vector that is perpendicular to the hyperplane and $\bar{u}$ be a vector to an unknown sample, as shown in Figure 3.6. The projection of $\bar{u}$ onto $\bar{w}$ is calculated with a dot product and compared with a threshold, as shown in Equation 3.5.20. With the threshold, b, and $\bar{w}$ it is possible to classify the unknown sample.

$$\bar{w} \cdot \bar{u} + b \geq 0 \begin{cases} then\ Class\ B \\ else\ Class\ A \end{cases} \tag{3.5.20}$$

3.5.4.4 Sample Classification Computational Complexity Study

The memory and computational complexity of the support vector machine depend on the kernel used. The linear support vector machine is similar to the logistic regression classifiers in terms of memory use and computational complexity, with the memory required being $8d$ bytes and the computational complexity to evaluate an unknown sample being $\mathcal{O}(d)$. Memory and computational requirements increase with the number of support vectors used in the classifier. It can be shown that the memory required for a classifier using a kernel is $8dn_{sv}$ bytes with n_{sv} being the number of support vectors used. The computational complexity of a classifier using a kernel is $\mathcal{O}(n_{sv}d)$ and can usually be simplified to $\mathcal{O}(d^2)$.

3.5.4.5 Advantages and Disadvantages

The advantages of the support vector machine approach are that the use of kernel functions makes it possible to transform linearly inseparable data into linearly separable data that can be classified. Support vector machines are defined by a convex optimisation problem, removing the risk of converging on local minima. The disadvantages of the

support vector machine are that the solving of the optimisation problem for the classifier training is computationally intensive, which means that training the classifier on large data sets may require a lot of computational resources. The optimisation step might not be able to converge to an answer. The selection of the correct kernel may require a large amount of human input that might be considered as a disadvantage.

3.5.4.6 Scikit-Learn Training Configuration Parameters

The Scikit-learn training configuration parameters are,

- A penalty parameter on misclassified data
- The kernels used
- Multiple kernel parameters
- The maximum number of iterations used for training.

A penalty parameter on misclassified samples can be adjusted to force a tight fit to the training data. Over-fitting might appear for large penalties on misclassified data.

Kernels are used to transform the samples into a different sample space. These four kernel functions were investigated for the upset detection problem,

- Linear kernel

$$K(\bar{x}_i, \bar{x}_j) = \bar{x}_i^T \bar{x}_j$$

- Polynomial kernel

$$K(\bar{x}_i, \bar{x}_j) = (\gamma \bar{x}_i^T \bar{x}_j + r)^d$$

- Radial basis function

$$K(\bar{x}_i, \bar{x}_j) = e^{(-\gamma |\bar{x}_i - \bar{x}_j|^2)}$$

- Sigmoid kernel

$$K(\bar{x}_i, \bar{x}_j) = \tanh(\gamma \bar{x}_i^T \bar{x}_j + r)$$

These kernels are commonly used in classification with support vector machines, and were therefore tested for the given upset detection problem. It is possible to specify a custom kernel specific to the problem. Custom kernels can influence the performance of the classifier drastically and may require a vast amount of fiddling to achieve good results. The focus of this project was to investigate the feasibility of classifiers for upset detection, and custom kernels were therefore considered to be outside of the scope of this project.

Different parameters within the kernel functions can be adjusted to shape the kernel function. Parameters that can be adjusted are γ, r and d, depending on the kernel function. The maximum iterations for the optimisation are also adjustable [30].

3.5.5 Decision Tree

3.5.5.1 Overview

A decision tree (DT) is a learning method which constructs a tree of decisions based on training data. The fitted classification model consists of a recursive tree of rules [31]. These rules are easily visualised and interpreted by a human. Figure 3.8 shows the decision boundary for a two-dimensional classification problem, and Figure 3.9 shows its corresponding tree.

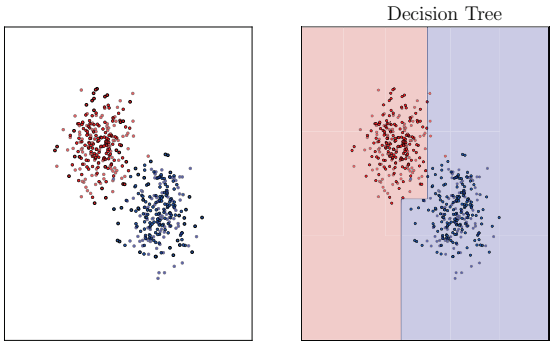

Figure 3.8: Two-dimensional data classified with a decision tree. The decision boundary is shown as the blue line in the figure.

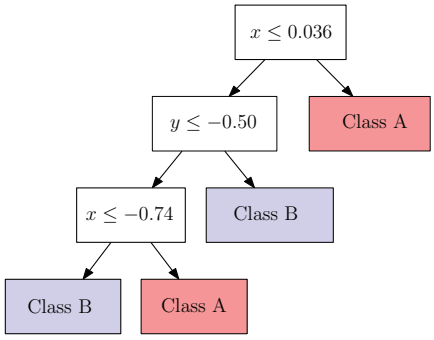

Figure 3.9: The decision tree that generated the decision boundary in Figure 3.8.

3.5.5.2 Training

The Scikit-Learn library uses the CART (Classification and Regression Tree) algorithm to construct decision trees. The CART algorithm generates binary splits and considers only one feature at a time. The splitting criterion is the information gain from the parent node to the two child nodes. The information gain is the difference in the impurity degree of the parent node and the child nodes after a split is made. The impurity degree is a measure of the homogeneity of the variables within a set, and the impurity degree shows the quality of the split made. The CART algorithm uses entropy or the Gini index as impurity degree measures to calculate the information gain [32]. Information gain is shown in Equation 3.5.21 with $L = \{left\ node, right\ node\}$ for classes A and B,

$$IG(S) = \text{Impurity of parent} - \sum_{i \in L} \frac{n_{A_i} + n_{B_i}}{n_A + n_B} \text{Impurity of child}(i). \tag{3.5.21}$$

where $n_{()}$ is the number of samples for a given class and node.

The entropy for class A and class B is shown in Equation 3.5.22, with $p(A)$ the probability of class A,

$$H(p(A), p(B)) = -p(A)\ln(p(A)) - p(B)\ln(p(B)). \tag{3.5.22}$$

The Gini index for class A and class B is shown in Equation 3.5.23, with $p(A)$ the probability of class A,

$$G(p(A), p(B)) = 1 - (p(A)^2 + p(B)^2) \tag{3.5.23}$$

.

The information gains for the two impurity degree measures are shown in Equations 3.5.24 and 3.5.25

$$I(S) = H(\frac{n_A}{n_A + n_B}, \frac{n_B}{n_A + n_B}) - \sum_{i \in L} \frac{n_{A_i} + n_{B_i}}{n_A + n_B} H(\frac{n_{A_i}}{n_{A_i} + n_{B_i}}, \frac{n_{B_i}}{n_{A_i} + n_{B_i}}) \tag{3.5.24}$$

$$I(S) = G(\frac{n_A}{n_A + n_B}, \frac{n_B}{n_A + n_B}) - \sum_{i \in L} \frac{n_{A_i} + n_{B_i}}{n_A + n_B} G(\frac{n_{A_i}}{n_{A_i} + n_{B_i}}, \frac{n_{B_i}}{n_{A_i} + n_{B_i}}) \tag{3.5.25}$$

The split, S, used to calculate the information gain, is a binary split on a single feature. Features are randomly selected, and the split values on the feature are selected at fixed intervals or at random values within the feature's range. The split with the maximum information gain is used, and the process is repeated for the next layer of the tree [33].

The tree is fully grown using the information gains at different splits. The grown tree is then pruned to reduce over-fitting, thus improving the generalisation of the tree. The pruning is done on predefined stopping criteria, like maximum tree depth, minimum number per leaf, minimum samples per split or maximum number of leaf nodes [33].

3.5.5.3 Classification

The tree of rules is applied to the unknown sample. The rules classify the sample into one of the two classes.

3.5.5.4 Sample Classification Computational Complexity Study

The memory required by the decision tree grows with the number of splits made in the tree, and is proportional to the depth of the tree. The maximum amount of memory used by the tree is dependent on the depth of the tree, $8(2^{(Tree\ Depth)} - 1)$ bytes. This amount of memory is for a symmetrical tree, which is the worst case. If the tree is asymmetrical that will result in less memory being used. The absolute worst computational complexity for a symmetrical decision tree is $\mathcal{O}(2^{(Tree\ Depth)})$, and the mean complexity for a symmetrical tree is $\mathcal{O}(\log 2^{(Tree\ Depth)})$. If the decision trees are not symmetrical, that will result in a less complex classifier.

3.5.5.5 Advantages and Disadvantages

The advantages of decision trees are that they are easily interpreted by humans, making them an intuitive way to classify data. A decision tree translates to a sequential list of if-statements that can be easily implemented on a computer. The major disadvantage of a decision tree is that it is prone to over-fitting, if it is insufficiently pruned.

3.5.5.6 Scikit-Learn Training Configuration Parameters

The Scikit-learn training configuration parameters are,

- The type of impurity degree measure used
- The maximum number of features used

- The maximum depth of the tree
- The minimum samples per leaf
- The minimum samples to split an internal node.

The impurity degree measures used by the Scikit-Learn library as splitting criteria are Gini index and entropy. The maximum number of features used for the training of the classifiers can be selected, with the default being all the features. The pruning parameters available to reduce over-fitting are the maximum depth of the tree, the minimum samples per leaf, the minimum number of samples required to split an internal node, and the maximum number of leaf nodes.

3.6 Ensemble Classification

Ensemble classification combines a number of classifiers into a single classifier. Over-fitting is reduced with ensemble classification, through the use of multiple classifiers to classify the same sample. Ensemble classifiers use majority vote, weighted average or a mean to combine the output of multiple classifiers. The classifiers can be trained on varying versions of the training data, or different classifiers can be trained on the same training data. The ensemble classifiers investigated in this study are shown in Table 3.4,

Table 3.4: Ensemble classifiers investigated.

Classifier Name	**Classifier Characteristics**
AdaBoost (Boost)	Classifiers trained on boosted sets of training data
Random Forest (RF)	Multiple decision trees trained on subsets of training data
Bagging Classifier (Bag)	Classifiers trained on subsets of training data
Multi-Classifier System (MCS)	A variety of classifiers combined into one classifier

3.6.1 AdaBoost Classifier

3.6.1.1 Overview

An AdaBoost classifier fits a sequence of weak learners (low complexity), which performs just better than random guessing, on repeatedly modified versions of the data. The predictions are combined through a weighted majority vote to produce the final prediction. The modifications consist of applying weights to each training sample. The initial weak learner is trained on the original data. For the successive iterations, the weights are individually modified to emphasise the missed detections [34].

Figure 3.10 shows the decision boundary for the two-dimensional classification case, using an AdaBoost classifier. The Adaboost classifier in Figure 3.10 used a decision tree as its base classifier.

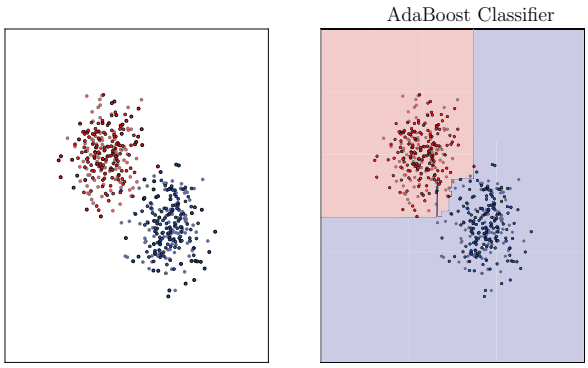

Figure 3.10: Two-dimensional data classified with an AdaBoost classifier using a decision tree as the base classifier. The decision boundary is shown as the blue line in the figure.

3.6.1.2 Training

Figure 3.11 shows the training process for the AdaBoost classifier using a linear SVM as the base classifier. This training scheme reduces the chance of over-fitting, by using a collective of classifiers.

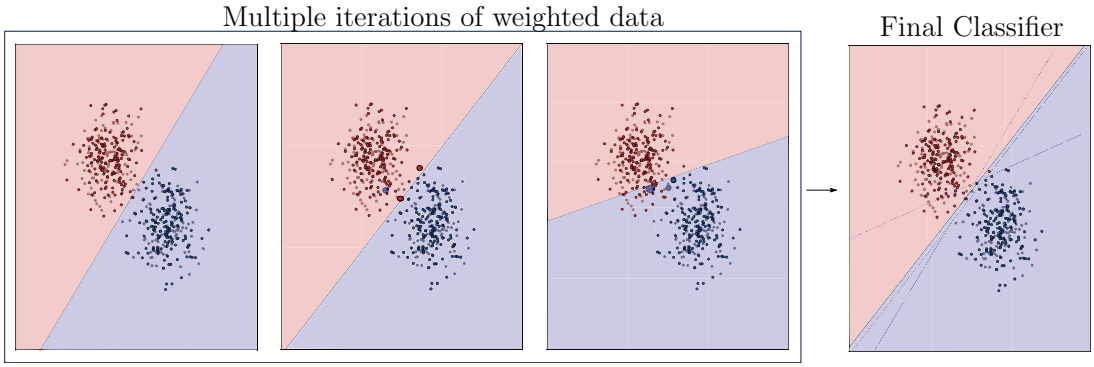

Figure 3.11: The process of training an AdaBoost classifier using a linear SVM classifier as the base classifier. The decision boundaries for the different iterations with weighted samples are shown.

Let the label y_i of a sample be in the labelset $Y = \{-1, +1\}$ and the weights of the training samples be denoted as $w_i^{(m)}$. The AdaBoost algorithm calls the base learning algorithm $(h^{(m)}(\bar{x}_i))$ repeatedly in a series of rounds $m = 1, \ldots, M$. The performance of the weak learner is measured by its error, as shown in Equation 3.6.1. A weak learner is expected to perform just better than random guessing, $err^{(m)} < 0.5$. The AdaBoost algorithm is shown in Algorithm 1.

$$err^{(m)} = \sum_{i:h^{(m)}(\bar{x}_i) \neq y_i}^{max} w_i^{(m)} \tag{3.6.1}$$

It is clear from the weak learner weight update in Algorithm 1 that $\alpha^{(m)} > 0$ and that $\alpha^{(m)}$ gets larger as $err^{(m)}$ gets smaller.

Algorithm 1 AdaBoost Training Algorithm [35]

1. Initialise the observation weights $w_i^{(1)} = \frac{1}{n}, i = 1, 2, \ldots, n$
2. For m = 1 to M
   a) Fit classifier, $h^{(m)}(\bar{x}_i)$ to the training data using weights $w_i^{(m)}$
   b) Compute error:
   $$err^{(m)} = \sum_{i:h^{(m)}(\bar{x}_i) \neq y_i} w_i^{(m)}$$
   c) Compute base classifier weight:
   $$\alpha^{(m)} = \frac{1}{2} \ln(\frac{1 - err^{(m)}}{err^{(m)}})$$
   d) Compute weight normalisation factor $Z^{(m)}$:
   $$Z^{(m)} = 2\sqrt{err^{(m)}(1 - err^{(m)})}$$
   e) Compute next step observation weights:
   $$w_i^{(m+1)} = \frac{w_i^{(m)}}{Z^{(m)}} e^{-\alpha^{(m)} y_i h^{(m)}(\bar{x}_i)}$$
3. Output of the final classifier:
   $$H(\bar{x}) = sign(\sum_{m=1}^{M} \alpha^{(m)} h^{(m)}(\bar{x}_i))$$

3.6.1.3 Classification

The output of the final classifier and can be taken as the weighted majority vote of all the trained classifiers in Equation 3.6.2,

$$H(\bar{x}) = sign(\sum_{m=1}^{M} \alpha^{(m)} h^{(m)}(\bar{x}_i)). \tag{3.6.2}$$

The base classifiers, $h^{(m)}(\bar{x})$, classify the data, as previously discussed.

3.6.1.4 Sample Classification Computational Complexity Study

The memory usage and computational complexity of the AdaBoost classifier are dependent on the memory requirements and computational complexity of the base classifier as well as the number of base classifiers, M, used. The memory required is therefore $M \times$ (Memory of base classifier) and the complexity is $\mathcal{O}(M$(Complexity of base classifier)).

3.6.1.5 Advantages and Disadvantages

The advantage of the AdaBoost classifier is that it generalises well, reducing the change of over-fitting. The disadvantage is that the computational complexity might, however, be high for complex base classifiers.

3.6.1.6 Scikit-Learn Training Configuration Parameters

The Scikit-learn classifier training configuration parameters are,

- The boosting training algorithm used
- The base classifier used
- The number of base classifiers used.

The two boosting training algorithms available are the SAMME and SAMME-R (Stage-wise Additive Modelling using a Multi-class Exponential loss function) algorithms [36]. SAMME and SAMME-R are multi-class boosting algorithms. The SAMME algorithm is the same as Algorithm 1 for binary classification. The SAMME-R algorithm uses weighted probability estimates to update the additive model, rather than the classifications themselves.

The base classifier can be selected; in this case decision tree classifiers were used as base classifiers. Training the decision trees is fast compared to a SVM, and the classification is computationally inexpensive. AdaBoost classifiers perform best when constructed with simple learners, like shallow decision trees. The number of classifiers in the ensemble can be selected.

3.6.2 Random Forest

3.6.2.1 Overview

The random forest classifier (RF) is an averaging of randomised tree algorithms using a perturb-and-combine technique specifically designed for trees. A diverse set of classifiers is created by introducing randomness into the classifier training. Figure 3.12 shows the decision boundary of the random forest for a two-dimensional classification case.

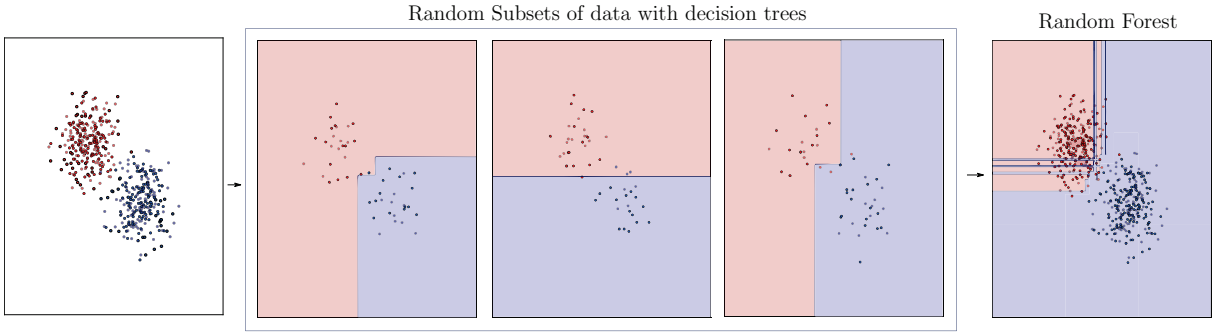

Figure 3.12: Two-dimensional data classified with a random forest classifier. The decision boundary is shown as the blue line in the figure. Some trained decision trees on subsets of the data are shown.

3.6.2.2 Training

A random forest classifier fits a number of decision trees on various subsets of the training data set[37]. These subsets are subsets of the feature space and the training samples. The split of a node is picked based on the best split within the subset of features, resulting in a slightly biased classifier. Averaging of the classifiers compensates for this bias. Trained decision trees, on subsets of the data and the features of the data, are shown in Figure 3.12.

3.6.2.3 Classification

An unknown sample is classified by each of the trained decision tree classifiers. The decision tree class outputs are combined by averaging the class prediction of the decision trees.

3.6.2.4 Sample Classification Computational Complexity Study

A random forest is an ensemble of trees. The memory usage of the random forest is therefore the product of the number of trees in the forest and the memory per tree. It can be shown that the maximum memory required by the random forest is $8M(2^{(Tree\ Depth)} - 1)$ bytes. The maximum complexity of the forest is scaled with the trees within the forest, $\mathcal{O}(M2^{(Tree\ Depth)})$ at the worst and $\mathcal{O}(M \log 2^{(Tree\ Depth)})$ as the mean maximum complexity.

3.6.2.5 Advantages and Disadvantages

The advantages of random forest classifiers are that they handle large training data sets considerably well, due to the bagging of the training data. The classifiers do not require linear features or even features that behave linearly. The main disadvantage of random forests is that they tend to over-fit on noisy training data.

3.6.2.6 Scikit-Learn Training Configuration Parameters

All the configuration parameters for the decision trees are available for this classifier. The number of classifiers can be selected, with the default being ten decision trees. Bootstrap samples are used when building the tree, but can be disabled.

3.6.3 Bagging Classifier

3.6.3.1 Overview

A bagging (bag) classifier trains classifiers on random subsets or "bags" of the data [38]. All the features are used to train the base classifiers. The bagging classifier combines the individual decision tree classifiers' predictions by averaging them [39]. Figure 3.13 shows the decision boundary for a bagging classifier containing ten decision trees.

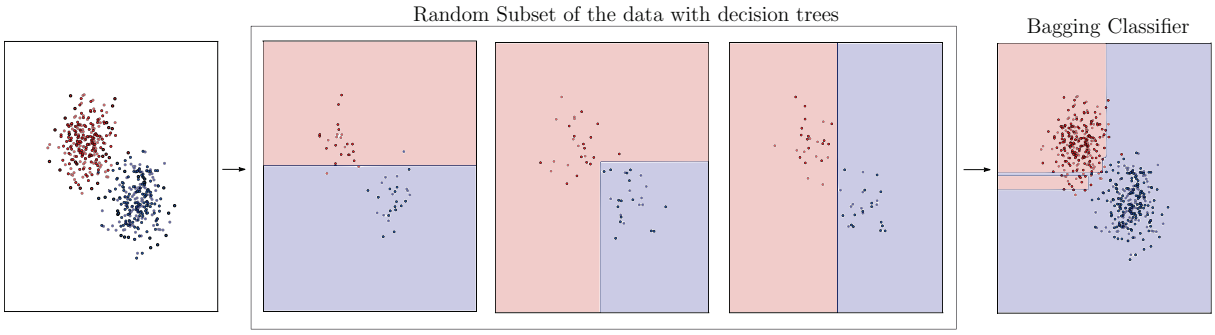

Figure 3.13: Two-dimensional data classified with a bagging classifier. The decision boundary is shown as the blue line in the figure. Some trained decision trees on subsets of the data are shown.

3.6.3.2 Training

This process is shown in Figure 3.13. The subset samples are drawn with replacement, known as bootstrapping. Multiple base classifiers are trained on the subsets of the training data. The trained base classifiers are aggregated into one ensemble classifier. Aggregation can be a majority vote or average of the predicted classes. No weights are assigned to the different base classifiers.

3.6.3.3 Classification

The classification process of the bagging classifier is the average of the ensemble base classifiers or the majority vote of the different base classifiers. The classification done by the base classifiers is exactly the same as what was described in the previous section for the non-ensemble classifiers.

3.6.3.4 Sample Classification Computational Complexity Study

The computational complexity and memory required by the bagging classifier are similar to those of the AdaBoost classifier. The memory and computational requirements are equal to the memory and computational complexity of the base classifier scaled with the number of classifiers in the ensemble.

3.6.3.5 Advantages and Disadvantages

The advantage of the bagging classifiers is that over-fitting is reduced by using classifiers trained on subsets of the whole training data set. Simple base classifiers can be combined to form a classifier with high accuracy. The disadvantage of bagging classifiers is the increase of the computational complexity as the product of the complexity of the base classifier and the number of base classifiers used. A bagging classifier may require large amounts of computational resources if a complex base classifier is used.

3.6.3.6 Scikit-Learn Training Configuration Parameters

The Scikit-learn classifier training configuration parameters are,

- The type of base classifier used
- The number of the base classifiers used

- The number of features and samples drawn into the subsets.

The type and number of base classifiers can be changed, with the default being ten decision tree classifiers. The number of samples and features drawn into the subsets can be specified with the option of bootstrapping on the samples and features.

3.6.4 Multi Classifier System

3.6.4.1 Overview

A multi-classifier system (MCS) is a system that consists of multiple classifiers and a fusion scheme to combine their predictions [40]. A MCS is analogous to a company's board of directors, where the board consists of people with different levels of qualifications and expertise. Adding diversity of different learning paradigms improves the overall performance of the classifier. The chance of over-fitting is also reduced. Figure 3.14 shows the decision boundary for three base classifiers, as well as the decision boundary of the MCS.

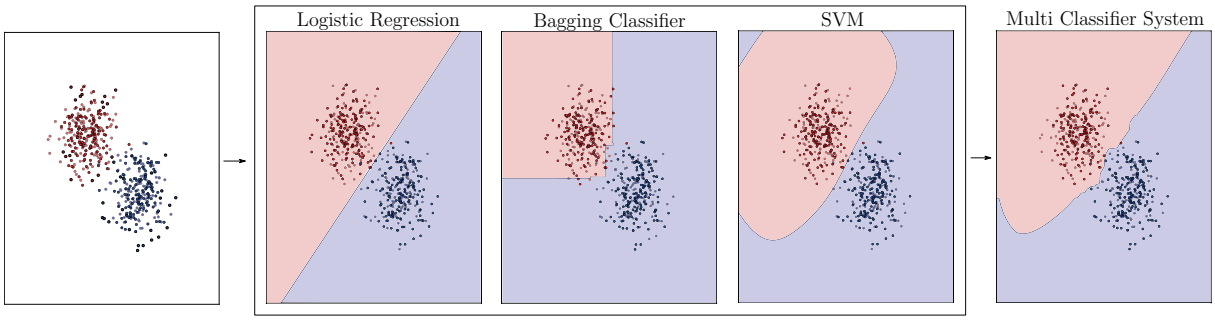

Figure 3.14: Two-dimensional data classified with a multi-classifier system containing a logistic regression model, a bagging classifier and a SVM. The decision boundary is shown as the blue line in the figure. The decision boundaries for the three base classifiers are shown.

3.6.4.2 Training

The base classifiers are trained each according to their own training method, then combined by taking the average of the predicted probabilities. Three classifiers relying on different assumptions were combined in this project, since this offered good generality while keeping computational resources low.

3.6.4.3 Classification

The base classifiers each perform their classification according to their base type. The three predictions are combined by taking the average of the predicted class probabilities. A probability greater than 0.5 is considered as upset and less than 0.5 is non-upset.

3.6.4.4 Sample Classification Computational Complexity Study

The memory required and the computational complexity of the multi-classifier system are simply the sum of the memory and computational resources required by all the base classifiers. The memory footprint is therefore $\sum$(Memory of base classifier) and the complexity is $\mathcal{O}(\sum$(Complexity of base classifier)).

3.6.4.5 Advantages and Disadvantages

The advantage of combining multiple classifiers that rely on different assumptions results in a classifier with little chance of over-fitting. The greatest disadvantage of the multi-classifier system is the high amount of computational resources required to classify an unknown sample. The accuracy of the multi-classifier system can be affected by one inaccurate base classifier, due to the relatively small number of classifiers used.

3.6.4.6 Scikit-Learn Training Configuration Parameters

The type and number of base classifiers and the averaging weights for the base classifiers can be specified. Uniform weights are used as the default configuration.

3.7 Summary of Computational Complexity of Classifiers

Table 3.5 shows a summary of the memory usage and computational complexity of the different classifiers. The nearest neighbour classifier is the only classifier of which the complexity increases with the number of the training samples, N. The number of features used influences the complexity of the Naïve Bayes, logistic regression and support vector machine classifiers. These features are bounded by the number of sensor measurements used in the upset detection system. The complexity of the system therefore decreases for the case where the anemometric sensors are not available. A support vector machine's complexity is dependent on the number of support vectors used to describe the hyperplane separating the two classes. The complexity of the tree-based classifiers is dependent on the depth of the trees. An ensemble classifiers' complexity is the sum of all the base classifiers in the ensemble and there is an overhead complexity added by the aggregation of the base classifiers. The overhead is small in comparison with the complexity of the base classifiers and is therefore ignored.

The memory requirement of the trained classifiers is dependent on the same parameters as the computational complexity of the classifiers.

Table 3.5: Summary of the classifiers' memory requirements and computational complexity to classify an unknown sample.

Classifier	**Memory Usage**	**Classification Complexity**
knn	$\approx dN$	$\mathcal{O}(\log N)$
NB	$\approx d$	$\mathcal{O}(d)$
LR	$\approx d$	$\mathcal{O}(d)$
SVM	$\approx dn_{sv}$	$\mathcal{O}(n_{sv}d)$
DT	$\approx 2^{(Tree\ Depth)}$	$\mathcal{O}(\log 2^{(Tree\ Depth)})$
Boost with DT	$\approx M2^{(Tree\ Depth)}$	$\mathcal{O}(M \log 2^{(Tree\ Depth)})$
RF	$\approx M2^{(Tree\ Depth)}$	$\mathcal{O}(M \log 2^{(Tree\ Depth)})$
Bagging with DT	$\approx M2^{(Tree\ Depth)}$	$\mathcal{O}(M \log 2^{(Tree\ Depth)})$
MCS with LR, SVM, bagging	$\approx (d + dn_{sv} + M2^{(Tree\ Depth)})$	$\mathcal{O}(d + dn_{sv} + M \log 2^{(Tree\ Depth)})$

Chapter 4

Design of an Upset Detection System

This chapter presents the design and validation of an upset detection system for commercial passenger airliners that uses classification algorithms operating on a combination of anemometric and inertial sensor measurements. The system is an intelligent upset awareness system that automatically detects and identifies aircraft upset conditions by using all available on-board sensors. The objective of the system is to provide reliable awareness and recognition of upset conditions to the pilot so that the correct recovery actions can be taken.

The chapter is structured as follows: First, the requirements and constraints of the upset detection function are listed. Following the requirements, the conceptual design and proposed architecture of the function are presented. Following this, the strategy that was used to generate the training and testing data for the classifiers is described. Next, the classifier training procedure is discussed. The classifier evaluation scheme is presented, this includes capturing the classifier accuracies and determining the classifiers' sensitivity to training parameters. The classification accuracy of the upset detection functions are also compared to the accuracy of estimation-based approaches. Next, the locations of the false alarms and missed detections relative to the classification boundary are evaluated. Finally, the upset detection function validation with simulated upset manoeuvres using a representative flight dynamics model for a commercial passenger aircraft is presented.

4.1 System Requirements and Constraints

The following requirements and constraints were formulated for the upset detection system:

- The system should be able to detect upset accurately and with a low probability of a false alarm being generated in the normal flight domain.
- The system must use classification techniques to perform the upset detection.
- The system should perform the upset detection based on sensor data from the anemometric and inertial sensors typically available on commercial passenger aircraft.
- The system should be able to detect upset using only the inertial sensors when the anemometric sensors are not available.
- The angle of attack upset detection system must be verified with representative sensor noise and external wind disturbances.

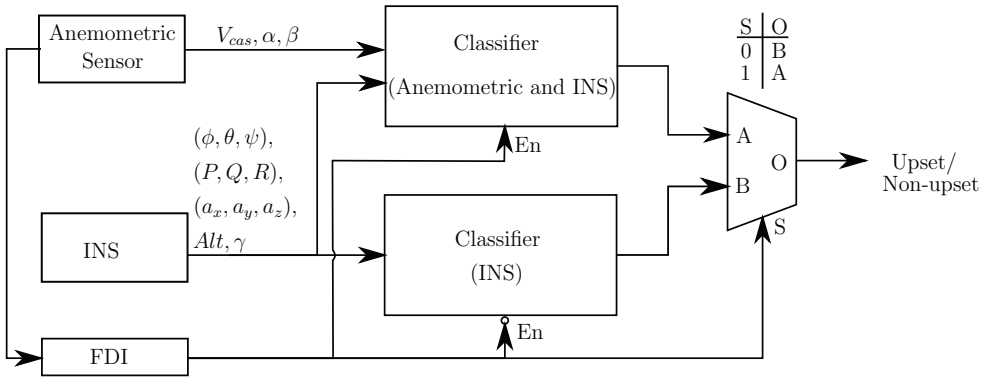

Figure 4.1: A system overview of the high angle of attack upset detection system

- False alarms should not occur in the normal flight domain, even amidst severe environmental conditions.

- The upset detection system must be validated with simulated upset manoeuvres (called "validation manoeuvres") using a representative flight dynamics model for a commercial passenger aircraft.

The training data for the classifiers is constrained to simulated data, as generating training data from actual flight upset events is not feasible. It is assumed that all the sensors are available, or are known to be faulty, it is assumed that fault detection and isolation is already performed at sensor level. The assumption is made that the inertial sensors are always available and reliable. It is assumed that the anemometric sensors may be lost, as anemometric sensors have a higher failure rate than inertial sensors, and a system that has some sort of redundancy is investigated.

4.2 Conceptual Design and Architecture

The conceptual design and architecture of the proposed upset detection function is shown in Figure 4.1. The upset detection function consists of two separate upset detection classification sub-systems. The primary classifier used both anemometric and inertial sensors to perform the classification, while the secondary classifier uses only the inertial sensors. It is assumed that a fault detection and isolation (FDI) system is implemented for the anemometric sensors. The FDI system is used to select between the two upset detection classifiers. When all the sensors are operational, Classifier A is enabled, and when faults are detected in the anemometric sensors, Classifier B is enabled.

The anemometric sensor measurements used by Classifier A are calibrated airspeed (V_{cas}), angle of attack (α) and side-slip angle (β).

The inertial sensor measurements used are the attitude angles (ϕ, θ, ψ), angular rates (P,Q,R), total body accelerations (a_x, a_y, a_z), altitude (Alt) and flight path angle (γ). Both the classifiers use the same inertial sensor variables as inputs. These anemometric and inertial measurements represent the sensor measurements that are typically available on a commercial passenger airliner.

The upset detection function outputs a binary signal that represents either upset or non-upset. The design and architecture presented in this section was followed for all upset classes.

4.3 Training Data Generation

The data set that was used for the classifier training and testing was generated from the Airbus A330 simulation model.

Initially it was thought that the time histories of simulated upset manoeuvres, where the aircraft is commanded to enter an upset condition, could be used to compile the training data set. However, it was found that in such a simulated upset manoeuvre, the aircraft spends most of the time in a non-upset condition, and only a very short time in the upset condition. A data set compiled from time histories of simulated upset manoeuvres would therefore contain many more examples of non-upset data points than examples of upset data points. The data set is consequently heavily biased towards non-upset data points, and a classifier trained on such a data set would therefore also have a bias towards classifying data points as non-upset conditions. Another disadvantage of a data set compiled from simulated upset manoeuvres it that it only covers the aircraft states that were reached by the specific upset manoeuvres that were performed, and will not necessarily cover the entire aircraft state space. A classifier trained on such a data set would therefore be unlikely to correctly classify aircraft states that are not close to those that were encountered in the simulated upset manoeuvres.

It was therefore decided to compile the data set with "snapshots" of randomly generated aircraft states and their associated anemometric and inertial sensor measurements. Using random "snapshots" instead of a manoeuvre trajectory is a valid approach, because the classification of an upset is based only on the current state of the aircraft, and not on the time history of its previous states. Although a "snapshot" of the aircraft state seems like a static condition, it does however represent a dynamic condition due to the fact that the state vector contains the translational velocities and angular rates of the aircraft as state variables. Compiling the data set from randomly generated aircraft states has several advantages. Firstly, the data set can be generated to contain an equal amount of non-upset data points and upset data points. The data set would therefore be unbiased, and a classifier trained on such a data set would consequently not be biased by the training data. Secondly, the data set can also be generated to cover the entire aircraft state space, and not just the states that were reached by specific upset manoeuvres performed. A classifier trained on such a data set could therefore be expected to correctly classify data points from the entire state space of the aircraft.

The approach was adopted to train and test the classifiers on a data set compiled of random "snapshots", but to then also validate the upset detection system in a dynamic aircraft simulation with simulated upset manoeuvres

The data set with "snapshots" of randomly generated aircraft states was compiled by generating each data point as follows:

1. The Airbus A330 simulation model is initialised with a random state, a random set of inputs, and a random set of atmospheric disturbances

2. The specific state variables that are randomly initialised are the aircraft's attitude (ψ, θ, ϕ), angular rates (P,Q,R), velocity vector (U,V,W) and the altitude. The specific inputs that are randomly initialised are the actuator deflections $(\delta_{elv}, \delta_{ail}, \delta_{rud})$, thrust command. The specific disturbance is the wind vector $(U_{wind}, V_{wind}, W_{wind})$

3. The sensor outputs from the anemometric and inertial sensors are calculated for the given states, inputs, and disturbances, and then representative sensor noise is added

4. The generated data point is labelled as either "non-upset" or "upset" by comparing it with the upset definitions.

A novel sampling strategy was used to generate the training and testing data set for the classifiers. The random samples were generated to be a union of a wide uniform sample distribution over the entire aircraft state space, and a tight normal distribution about the upset boundary. The wide uniform sample distribution enables the classifier to correctly identify non-upset and upset conditions that are far away from the upset boundary, while the tight normal distribution enables the classifier to accurately discriminate between non-upset and upset conditions close to the upset boundary.

4.4 Classifier Training

A number of different classifiers were trained on a data set to detect upset. The investigated classifiers were presented and discussed in depth in Chapter 3. The classifier training process was as follows,

- The complete data set was separated into a set of training data and a set of testing data, with 70% of the data set allocated for training and 30% of the data reserved for testing
- The different classifiers were all trained on the same set of training data that contained equal amounts of non-upset and upset data points.
- The classifiers were trained on data sets containing both anemometric and inertial sensor measurements, and on data sets containing only the inertial sensor measurements.
- The receiver operating characteristic (ROC) curves for the classifier were used to select a probability threshold, to ensure that the classifiers function at an equal error rate (EER) as discussed in Chapter 3. An example of a typical ROC curve is shown in Figure 3.1. The comprehensive set of ROC curves for all the classifiers can be seen in Appendices C to E.
- The learning curves of classifiers were determined and used to select the size of the training set to ensure that all classifiers were correctly trained, neither under-trained nor over-trained. The learning curves for the classifiers were determined by training the classifiers on data sets of increasing size and plotting the percentage correct detections against the number data points used.

Figure 4.2 shows an example of a learning curves for the high angle of attack upset detection multi-classifier system for both sensor cases. The comprehensive set of learning curves for all the different classifiers are shown in Appendices C to E. Learning curves are commonly used to determine whether classifier training data sets are large enough to ensure a sufficiently trained classifier. It can be seen from Figure 4.2 that the accuracy of the classifier eventually flattens out, and that the confidence interval decreases, as the training data set's size increases, which indicates that a classifier is adequately trained. For example a training data set of 5000 samples or greater would be sufficient for the MCS in Figure 4.2.

Ensuring that all the classifiers are adequately trained allows for an even comparison of classifier accuracies.

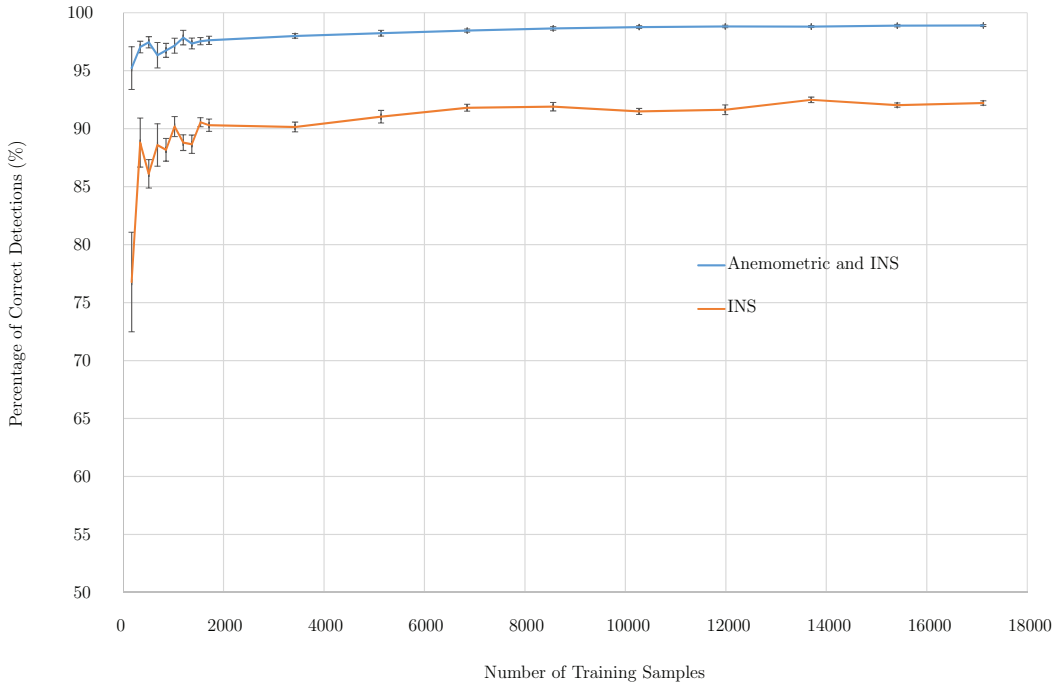

Figure 4.2: The multi-classifier system's learning curves for both sensor cases

4.5 Testing Classifier Accuracy

The different classifiers were then tested on the testing data set and their performances as classifiers were evaluated. The accuracy of the classifiers were evaluated in terms of total accuracy, $0.5 \times (\%\text{True Positives} + \%\text{True Negatives})$, since the classifiers operated at an EER which implies equal percentages of false alarms and missed detections.

Ten-fold Monte Carlo cross-validation was used to ensure that the classifier accuracy results are independent from how the data set was partitioned into training data and testing data. The complete data set was randomly separated into 70% training data and 30% testing data, but with the constraint that both data sets must contain equal amounts of non-upset and upset data points. The classifier was then trained on the random set of training data and tested on the random set of testing data. The training and testing cycle was repeated ten times with random partitions each time, and then the average accuracy of each classifiers over all ten cycles was calculated.

Each fold of the ten-fold Monte Carlo cross-validation was repeated ten times for all the non-deterministic classifiers. This ensured that the classifier accuracy results are independent from how the classifier training algorithms are initialised. The average accuracy of each classifier was taken as the average for the combined repetitions of the ten-fold cross-validation and the ten repetitions per cross-validation fold. The 95% confidence intervals for all the accuracies shown were calculated with the accuracies achieved by these repetitions.

A McNemar's test was used to determine the statistical significance of classifiers with overlapping confidence intervals. The test is used to determine whether the classifier test set is large enough to show that the differences in classifier accuracies are statistically

significant. Results were considered to be statistically significant for p-values smaller than 0.05.

The McNemar's test is shown by Equation 4.5.1 and Table 4.1. The discordants shown by cells b and c in Table 4.1 are used to determine $\mathcal{X}^2$ as shown in Equation 4.5.1. $\mathcal{X}^2$ has a chi-squared distribution with one degree of freedom. If $\mathcal{X}^2$ is significant ($p < 0.05$), the test dataset was considered as large enough to show the differences in classifier accuracies.

Table 4.1: McNemar's test - contingency table

	Classifier A - Non-Upset	**Classifier A - Upset**
Classifier B - Non-Upset	a	b
Classifier B - Upset	c	d

$$\mathcal{X}^2 = \frac{(b-c)^2}{b+c} \tag{4.5.1}$$

4.6 Classifier Sensitivity to Training Configuration Parameters

An investigation was performed to explore the sensitivity of the classifier accuracies to the training parameters that were used. For the tests performed in this chapter, the different classifiers were trained using the default training parameters provided by Scikit-learn. However, the training parameters can be tuned to optimise the classifier accuracy for the specific classification problem. Experiments were performed where the classifiers were trained with randomly initialised training parameters and then tested to determine the classifier accuracies for each set of training parameters. The investigation was performed to get an indication of the potential gain in classifier accuracy that could be obtained by tuning the training parameters for optimal classifier performance, and also to get an indication of the amount of tuning effort/expertise that would be required. The distribution of the classifier accuracies over the range of randomly initialised training parameters indicates the potential gain as well as the tuning effort/expertise required. The training parameters that were randomly varied, with their associated ranges are shown in Appendix B.

4.7 Comparison of Classification-based and Estimation-based Upset Detection

The classification accuracies of the different classifiers were compared to the accuracy of performing upset detection by simply thresholding a direct measurement or estimate according to the specific upset definition. This investigation was performed to evaluate the classification-based approach to upset detection proposed in this thesis against the more classical estimator-based approaches to upset detection that were found in the literature. To avoid having to physically implement all the different estimator-based approaches found in literature, the abstraction was used that the estimated anemometric variable provided by an unbiased state estimator could be modelled by adding zero mean Gaussian

white noise, with a variance equal to the estimate covariance of the estimator, to the true anemometric variable. The upset detection accuracies of the classification-based approach and the estimation-based approach were therefore compared as a function of the amount of noise added to the measurements/estimates of the anemometric variables. The objective was to determine at what level of measurement/estimation noise does the classification-based approach provide better accuracy than the estimator-based approach.

4.8 Evaluating Locations of False Alarms

The practical performance of the different classifiers as upset detection functions were evaluated. The anemometric measurements at which the false alarms and the missed detections occurred were compared to the classification boundary and also to the statistical distribution of the anemometric measurements that is expected to be encountered in normal flight with moderate turbulence. This was done to verify that the false alarms and missed detections only occur close to the classification boundary and that there are neither missed detections that are deep into the upset domain, nor false alarms that are well within the domain of normal flight. The locations of the false alarms generated by the upset detection system is an important metric to ensure a reliable system. The anemometric measurements at which false alarms occur should be located near the classification boundary to ensure sufficient separation between the false alarms and the anemometric measurements encountered in normal flight domain.

4.9 Validation Manoeuvres

Simulations were then performed with the Airbus A330 model to validate the performance of the upset detection function with simulated upset manoeuvres.

Validation manoeuvres are typically used to validate guidance and control law systems that are in the development stage. The aim is usually to check the compliance of the developed systems with the set of requirements. The set of manoeuvres used to evaluate longitudinal controllers was modified to make the aircraft enter the upset domain. Manoeuvres resulting in upset or recovering from upset were used to determine whether the classifiers were trained on representative data, as well as to verify the classifiers' accuracies in simulated flight.

Three types of validation manoeuvre were performed: within normal flight envelope, exiting normal flight envelope into the upset domain, and re-entering the normal flight envelope when recovering from the upset domain. The validation manoeuvres were performed at a range of flight points. These flight points were distributed as a grid at fixed intervals over the aircraft's airspeed and altitude range.

Chapter 5

High Angle of Attack Upset Detection

This chapter presents the definition of high angle of attack and the evaluation of an angle of attack upset detection system for commercial passenger airliners that uses classification algorithms operating on a combination of anemometric and inertial sensor measurements.

The chapter is structured as follows: First, a definition of high angle of attack upset is formulated to serve as the basis for the upset detection. Next, the training and testing data for the classifiers is described. A number of different classifiers are then trained on the data set to detect the angle of attack upset. The different classifiers are tested on the testing data set and their performances as classifiers are evaluated. As part of the evaluation, an investigation is performed to explore the sensitivity of the classifier accuracy results to the training parameters that were used. The classification accuracy of the classification-based approach is also compared to the accuracy of estimation-based approaches. Next, the locations of the false alarms and missed detections relative to the classification boundary are evaluated and compared to the angle of attack distribution encountered in normal flight. Finally, the angle upset detection system is validated with simulated upset manoeuvres using a representative flight dynamics model for a commercial passenger aircraft.

5.1 Definition of High Angle of Attack Upset

Before angle of attack upset detection can be performed, we must first define precisely what is considered to be a "high angle of attack upset" so that different aircraft states (and their associated sensor measurements) can be labelled as either "non-upset" or "upset".

The upset detection classifiers will be trained on the labelled data, which means that the final upset detection will be performed based on the upset detection that was used to label the data. In this study a high angle of attack upset was considered as an angle of attack exceeding the critical angle of attack. The critical angle of attack is the angle of attack at which the lift coefficient, as a function of angle of attack, reaches a maximum, as shown in Figure 5.1. The critical angle of attack was calculated as a function of altitude and airspeed. It was found that the altitude and airspeed were the predominant influence on the critical angle of attack, and were therefore used. This resulted in a surface that was a function of altitude and airspeed and that separates the upset and non-upset classes, as shown in Figure 5.2. The critical angle of attack surface for the Airbus A330 aircraft is shown in Figure 5.3. It can be seen that the critical angle of attack has a strong

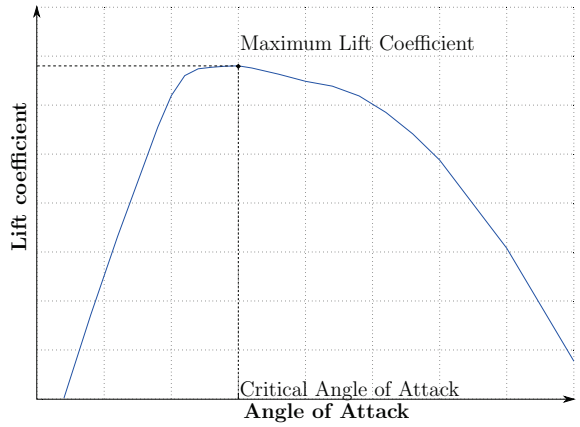

Figure 5.1: Lift generated as a function of angle of attack. The maximum lift point at the critical angle of attack is shown.

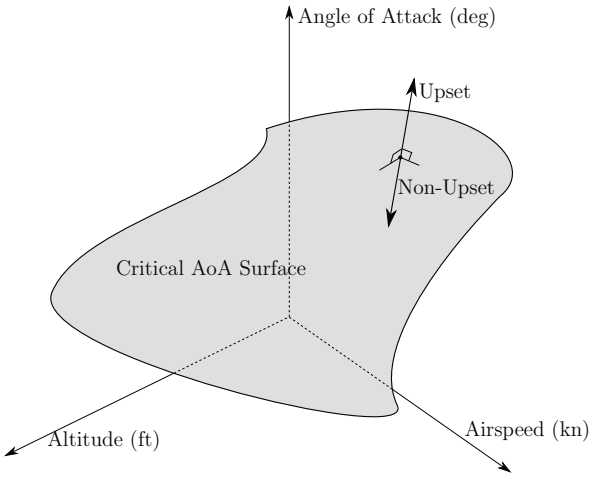

Figure 5.2: The critical angle of attack surface that separates the upset and non-upset classes.

dependence on the dynamic pressure (the combination of airspeed and altitude). The true angle of attack of the simulated aircraft relative this surface is used to label given data points.

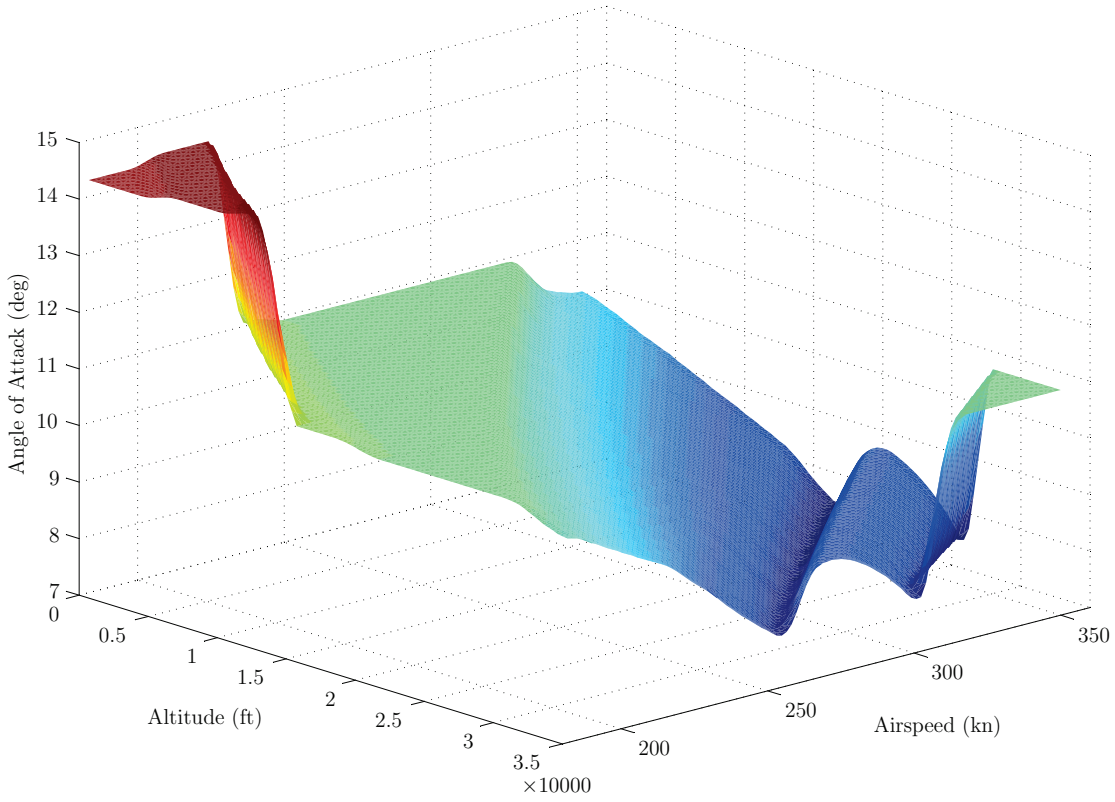

Figure 5.3: The critical angle of attack surface for the Airbus A330 aircraft.

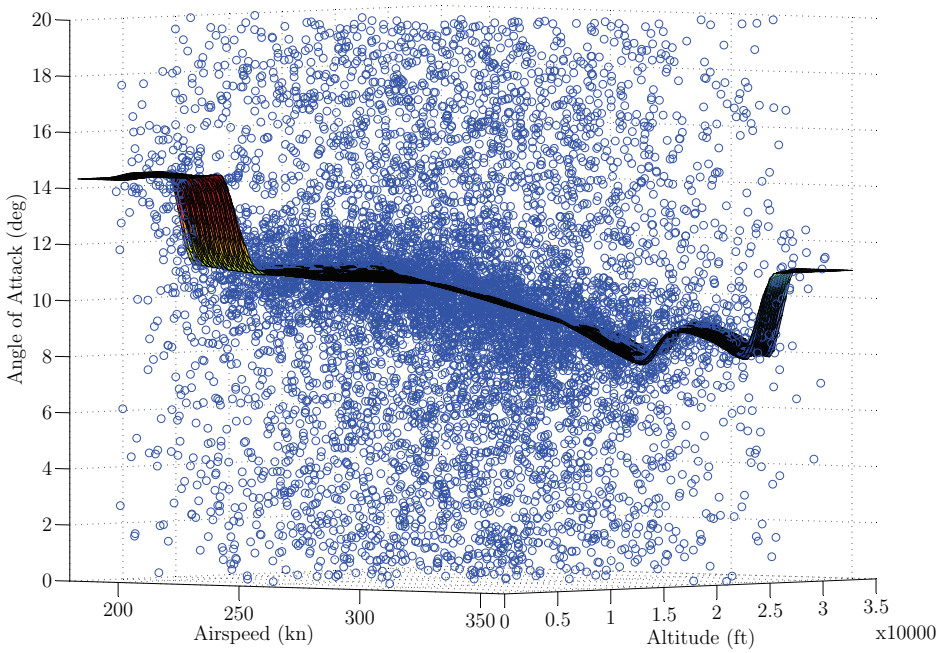

Figure 5.4: Random samples generated for the high angle of attack classifier training, with the critical angle of attack boundary also shown.

5.2 Training Data

Figure 5.4 shows the samples distributed about the high angle of attack upset boundary. The tight normal distribution of samples was chosen to have a standard deviation of 0.5°. A detailed description of the ranges and statistical distributions of all state variables, input variables, and disturbance variables used to generate random data points for the training data is given in Appendix A.

5.3 Classifier Results

This section gives the results of the tests conducted on the different classifiers as discussed in Chapter 4. The comprehensive set results for all the classifiers investigated can be seen in Appendix C.

5.3.1 Classifier Accuracy

Figure 5.5 shows the classifier accuracies of the different classifiers investigated, for both of the sensor cases. The 95% confidence intervals for the accuracies shown were within a 0.1% interval.

When using both anemometric sensors and inertial sensors, most of the classifiers provided classification accuracies greater than 96%, with AdaBoost providing the best accuracy of 99.23%, and Nearest Neighbour and Naive Bayes providing the worst accuracies of 84.6% and 89.24% respectively. Decision Trees were the top performing non-ensemble classifier with an accuracy of 98.28%.

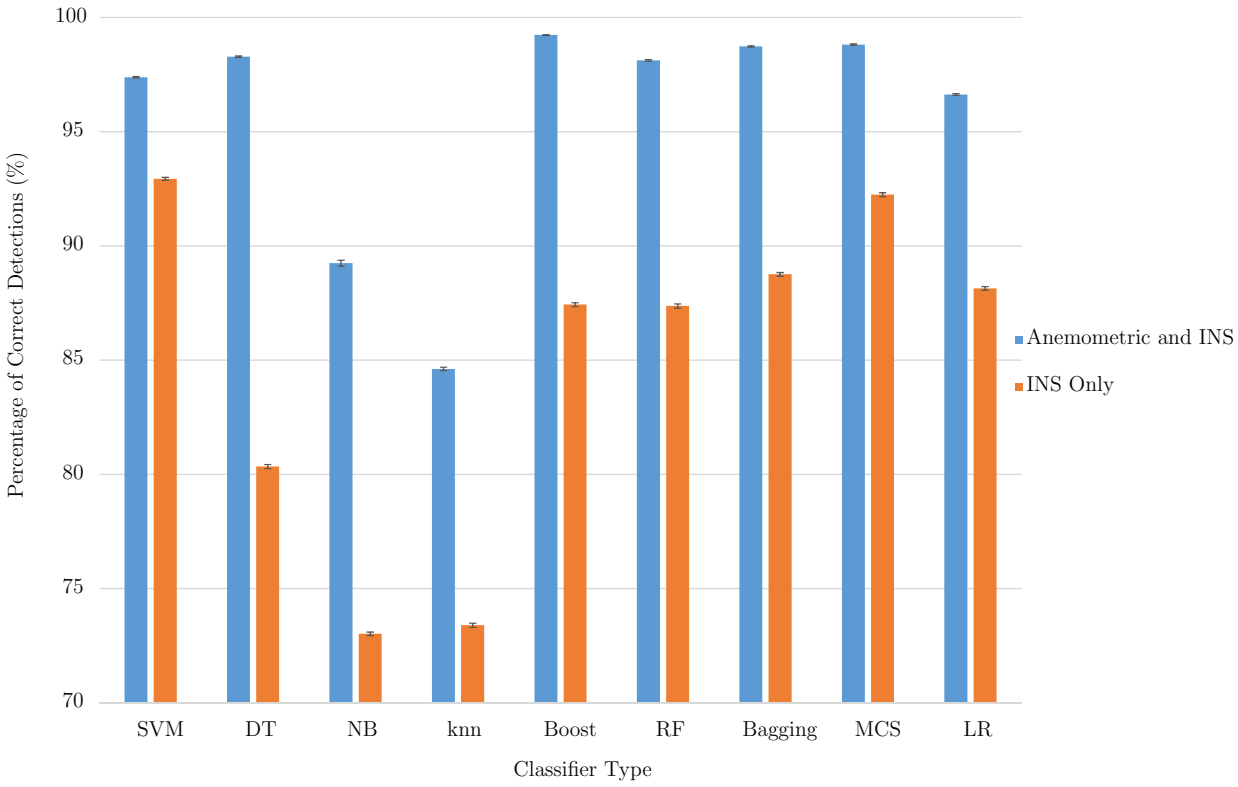

Figure 5.5: The percentage of correct detections for the different classifiers, for both sensor cases.

When using only inertial sensors, the classification accuracies of the classifiers dropped by about 10.8% on average, with the Support Vector Machine providing the best accuracy of 92.93% and the Naive Bayes providing the worst accuracy of 73.02%.

The AdaBoost and random forest classifier trained with only inertial sensor data were the only classifier with overlapping accuracy intervals. McNemar's test showed that the accuracies produced by the AdaBoost and random forest classifiers were statistically significant.

The majority of the classifiers had an accuracy higher than 96% for anemometric and inertial sensors available, with the naïve Bayes and nearest neighbour classifiers having accuracies below 96%. The high accuracies were expected due to the fact that all the information that describes the upset boundary was given to the classifiers. A model or approximation of the upset boundary was constructed in the training process.

Removing the anemometric sensor data resulted in a reduction of information supplied to the classifiers and caused a significant drop in classifier accuracy. An accuracy drop was expected as the classifiers needed to infer angle of attack information from the inertial sensors. The classifiers had no explicit information about the aerodynamics of the aircraft. The support vector machine and the multi-classifier system were able to classify high angle of attack upset with only inertial information with accuracies above 92%. This implies that it is possible to detect high angle of attack upsets without anemometric data, and

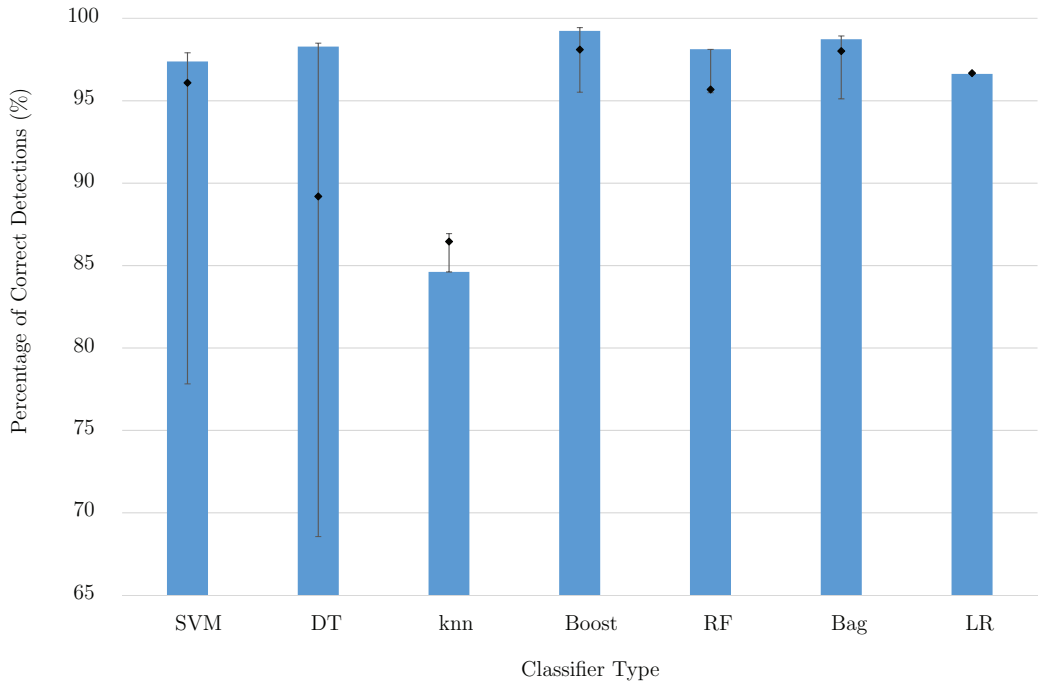

Figure 5.6: The default classifier configuration accuracies with the median, minimum and maximum accuracies achieved in the random search of the training parameters, for anemometric and inertial sensors available.

using only inertial data, with sufficient accuracy.

5.3.2 Sensitivity of Classifier Accuracy to Training Parameters

The default accuracy for the classifiers trained with anemometric and inertial sensors are shown in Figure 5.6, with the blue bars. The median, minimum and maximum accuracies of the random searches are shown with the black error lines. It was found that the support vector machine and the decision tree's searched parameter space produced a large accuracy interval. This shows that a small change in the classifier's training parameter might result in a large accuracy change, and therefore requires a great amount of human input to optimise. The nearest neighbour classifier was the only classifier where the default classifier training configuration produced the lowest accuracy. This shows that the nearest neighbour might greatly benefit from some training parameter optimisation. The rest of the classifiers' default accuracies were greater than the medians of the accuracies achieved with the parameter search. This shows that the default configurations were reasonable choices on which to evaluate the accuracies of the different classifiers for high angle of attack upset detection. The logistic regression classifier showed little change in accuracy with different training parameters.

The default accuracies and the accuracies obtained from the training parameter search for only the inertial sensors available are shown in Figure 5.7. The variation in the classifier accuracies achieved by the random training parameter search increased drastically with the exclusion of the anemometric data in the training data set, with the exceptions of the nearest neighbour and logistic regression classifiers. The classifiers' accuracies produced

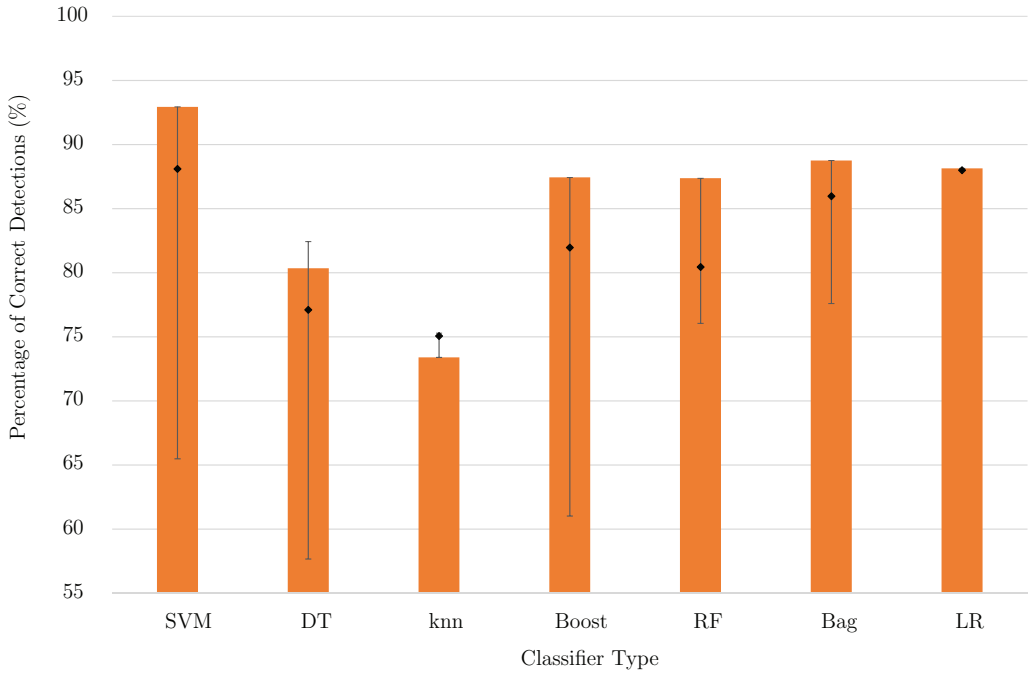

Figure 5.7: The default classifier configuration accuracies with the median, minimum and maximum accuracies achieved with the random search of the training parameters, for only inertial sensors available.

by the default configuration were in many cases the maximum accuracy found by the random search. The default configurations were therefore viable options for classifier evaluation with high angle of attack upset using only inertial sensors in mind.

The random search conducted showed that the default classifier configurations were reasonable choices on which to evaluate the accuracies of the different classifiers for high angle of attack upset detection. The default classifier configurations were therefore used throughout the tests conducted on the classifiers.

5.3.3 Classification vs. Estimation-based Upset Detection with Noisy Anemometric Sensors

Figure 5.8 shows the classification accuracy of the multi-classifier system (MCS) compared to the accuracy of simply thresholding a noisy angle of attack against the critical angle of attack. The classification accuracy of the MCS is shown for both sensor cases. The accuracy when using both anemometric sensors and inertial sensors decreased as the noise increased, and flattened out at 91%. The accuracy when using only inertial sensors remained constant at 92%. It was found that the classification-based approach outperformed the estimator-based approach when the standard deviation of the angle of attack measurement/estimation noise exceeded 0.55 degrees.

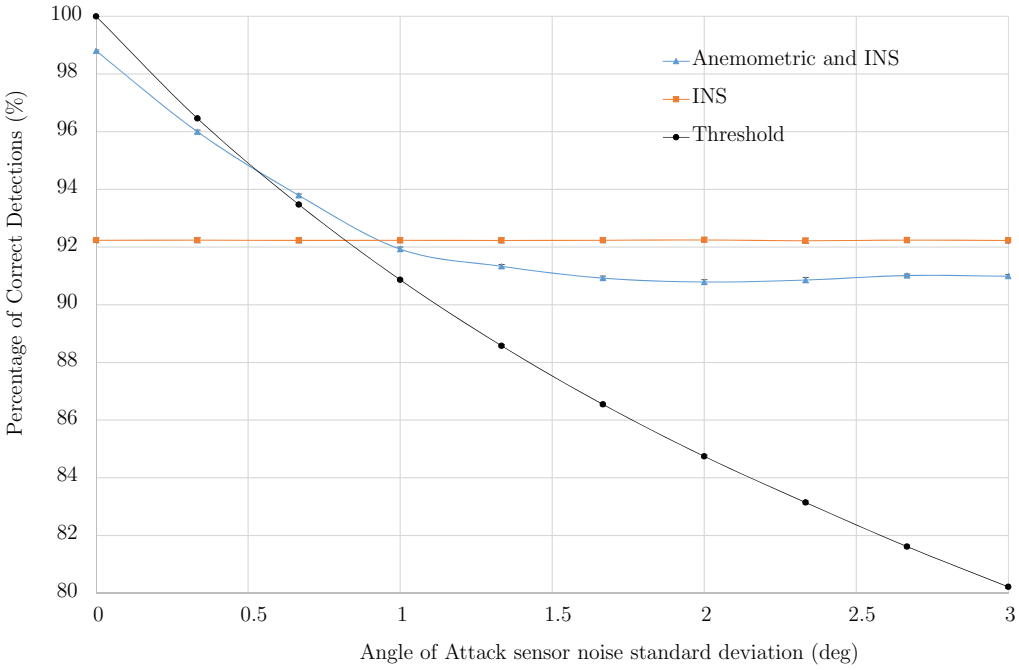

Figure 5.8: The MCS's accuracy at different angle of attack sensor noise levels for both sensor cases, compared with the threshold method.

5.3.4 Locations of False Alarms

Figure 5.9 shows the angle of attack distribution of the accumulated false alarms (shown in red) and the angle of attack distribution for an A330 in normal flight (shown in green). The angle of attack distributions shown were around the flight point of $240kn$ and $17500ft$. It was found that the angle of attack separation between the two distributions for the sensor case with the anemometric and inertial sensors available was 5.5°.

The angle of attach separation between the two distributions decreased to 4.5° with the exclusion of the anemometric sensor data, as shown in Figure 5.10. This exclusion resulted in a notable increase of false alarms situated around the flight point of $240kn$ and $17500ft$, from 0.65% to 10.19%.

The angle attack distribution of the false alarms was sufficiently removed from the angle of attack distribution of an A330 in normal flight to ensure a reliable upset detection system.

5.4 Simulation of Validation Manoeuvre

The elevator was oscillated between the minimum and maximum allowable deflections in order to generate the manoeuvre shown in Figures 5.11 and 5.12.

Figure 5.11 shows the false alarms and missed detections generated by the high angle of attack upset detection system with the anemometric and inertial sensors available. The angle of attack trajectory is shown in blue, showing that a combination of all three manoeuvres was evaluated within this trajectory. It is clear that all the false alarms

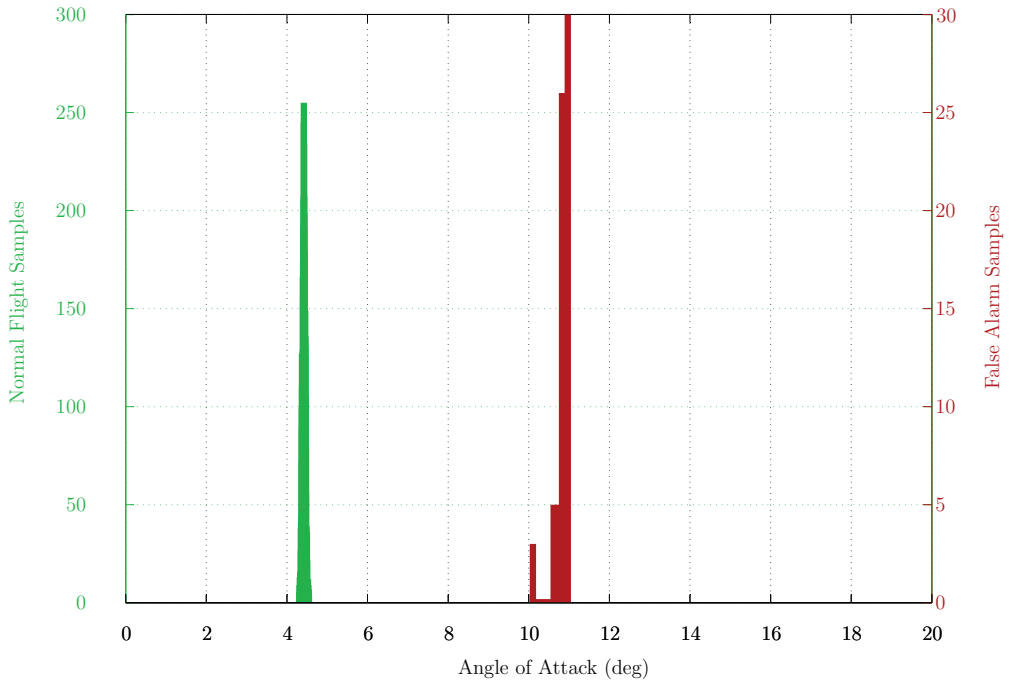

Figure 5.9: Angle of attack distribution for the false alarms generated by the MCS and the angle of attack distribution during cruise flight under moderate turbulence for anemometric and INS sensors available.

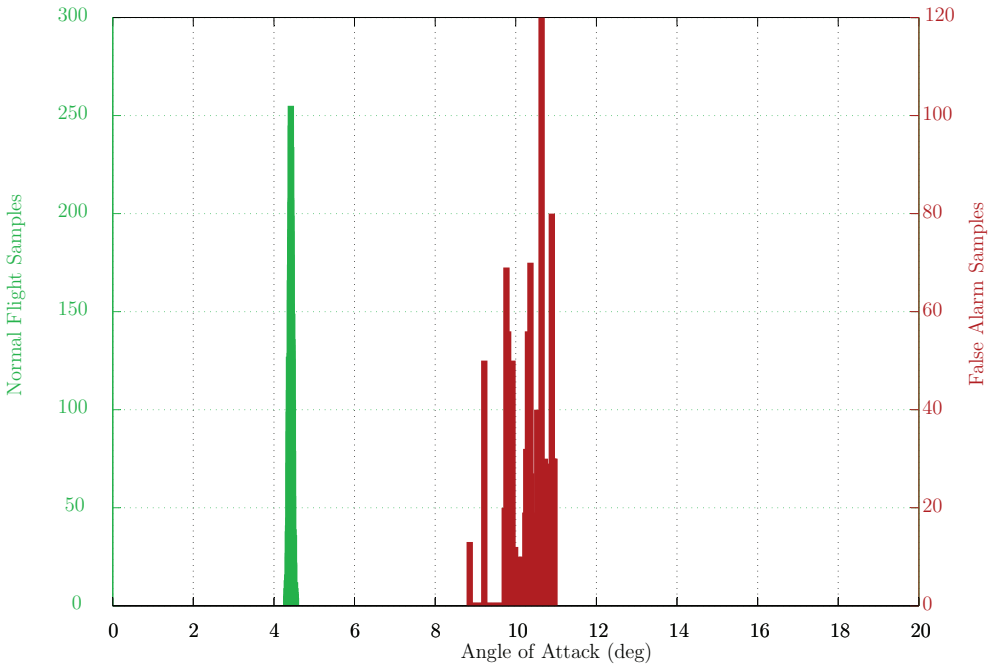

Figure 5.10: Angle of attack distribution for the false alarms generated by the MCS and the angle of attack distribution during cruise flight under moderate turbulence for only the INS sensors available.

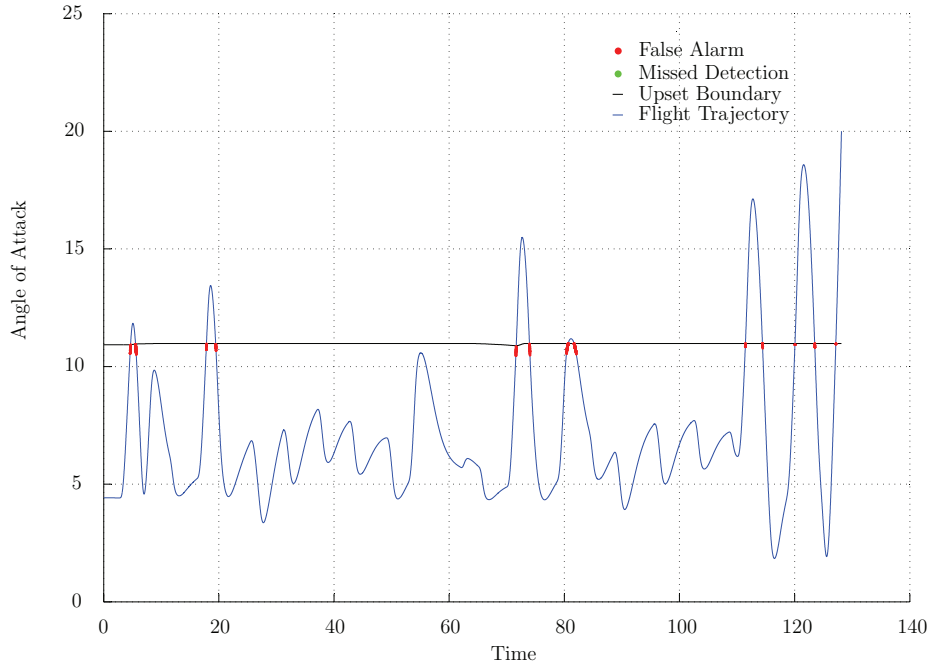

Figure 5.11: High angle of attack upset manoeuvre. The three types of manoeuvres are shown. The false alarms and missed detections by the classifier trained on the anemometric and inertial data are shown.

and missed detections generated by this system occur in close proximity to the upset boundary.

Figure 5.12 shows the false alarms and missed detections for the same flight trajectory but with upset detection performed using only inertial sensors. The number of false alarms generated by the MCS increased from 0.78% to 4.39% for the manoeuvre shown. The false alarm distance from the upset boundary increased, but remained within 3° from the upset boundary. The aircraft's trim angle of attack for this specific flight point was $\approx 4.5°$, the false alarms were still sufficiently removed from the typical angle of attack expected during normal flight. Missed detections were not produced for either of the two sensor cases.

The high angle of attack upset detection system was validated with flight manoeuvres using a representative simulation of a commercial passenger airliner. The results showed that the false alarms and missed detections lie in vicinity of the upset boundary as found for the random samples in the previous section. This shows that the proposed random training data generation method is an effective way to train classifiers for high angle of attack upset detection.

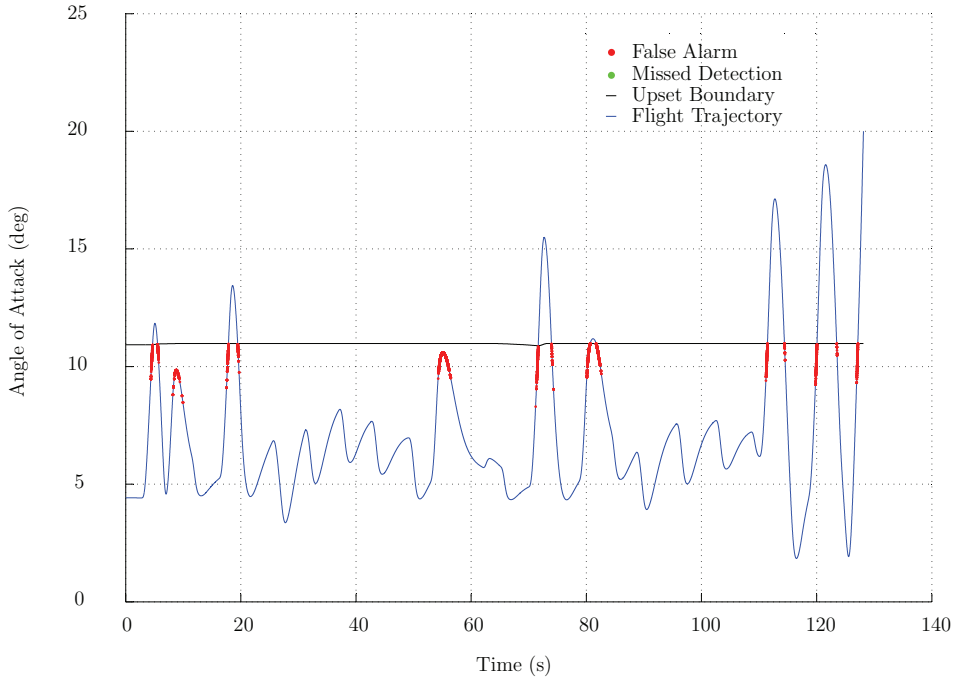

Figure 5.12: High angle of attack upset manoeuvre. The three types of manoeuvres are shown. The false alarms and missed detections by the classifier trained on only the inertial data are shown.

5.5 Conclusions

The majority of the classifiers had an accuracy greater than 96% when the anemometric and inertial sensors were used, with the false alarms in close proximity to the high angle of attack upset boundary. When using only the inertial sensors, the multi-classifier system had the best performance with an accuracy of 92.5% and the false alarms in close proximity to the high angle of attack upset boundary. The random search of the classifier training parameters showed that the default classifier configurations were a good basis for classifier comparison, as the default configuration produced an accuracy close to the maximum accuracy achieved by the random search. The classification-based approach for high angle of attack upset detection outperformed the estimator-based approach when the standard deviation of the angle of attack measurement/estimation noise exceeded 0.55 degrees.

Classification algorithms proved to be a viable option for high angle of attack upset detection. The multi-classifier system consisting of a support vector machine, bagging classifier and a logistic regression classifier had the best overall performance for high angle of attack upset detection, with the false alarms in close vicinity of the upset boundary and achieving reliable upset detection during the validation manoeuvres.

It was also found that high angle of attack upset can be detected without anemometric data, and using only inertial data, with sufficient accuracy.

Chapter 6

Underspeed Upset Detection

This chapter presents definition of underspeed and the evaluation of an underspeed upset detection function for commercial passenger airliners that uses classification algorithms operating on a combination of anemometric and inertial sensor measurements.

The chapter starts with a definition of underspeed upset, and a description of the classifier training data. Next, the total classifier accuracies and sensitivity to training parameters are shown. A comparison between the classification-based approach and estimation-based approach of upset detection is presented. Following this, the locations of the false alarms and missed detections from the upset boundary are evaluated. Finally, the underspeed detection system is validated with simulated upset manoeuvres using a representative flight dynamics model for a commercial passenger aircraft.

6.1 Definition of Underspeed Upset

Underspeed is defined as an airspeed that produces insufficient lift to support the aircraft's weight. The airspeed and the angle of attack of the aircraft are closely coupled. A reduction in airspeed results in an increase of angle of attack in order to produce sufficient lift. The underspeed boundary was calculated as the intersection between the trim (steady state) angle of attack and the critical angle of attack at different altitudes. Figure 6.1 shows the underspeed boundary used by the underspeed detection system. Samples with airspeeds lower than the underspeed airspeeds shown in Figure 6.1 were labelled as underspeed upset. It can be seen from Figure 6.1 that underspeed airspeed is proportional to altitude. This is caused by the higher airspeed required to maintain the same dynamic pressure due to the drop in air density at higher altitudes.

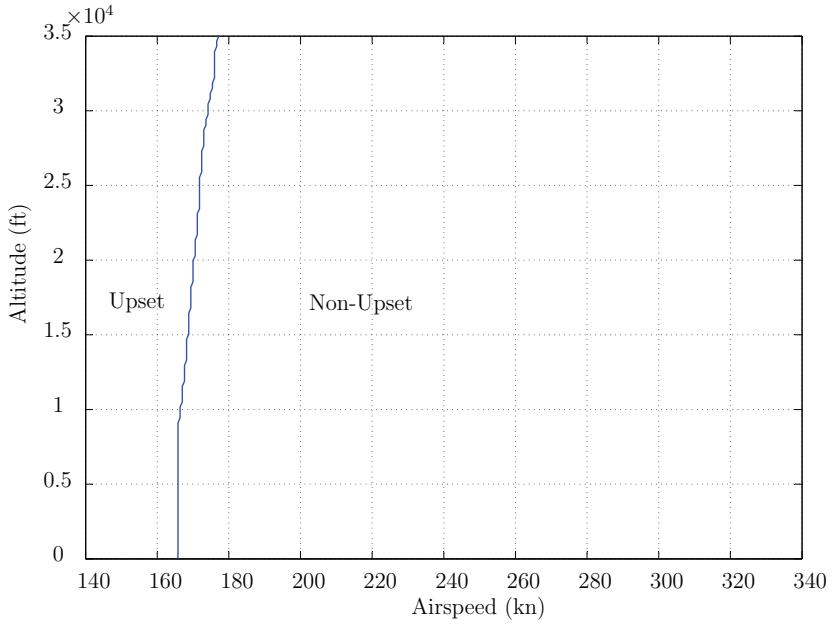

Figure 6.1: The underspeed boundary for an Airbus A330 aircraft

6.2 Training Data

The data set that was used for the classifier training and testing was generated from the Airbus A330 simulation model.

The tight normal distribution around the underspeed boundary had a standard deviation of $3.5kn$, as shown in Figure 6.2. The uniformly distributed samples were generated within the aircraft state and input ranges. The underspeed detection classifiers were trained on a data set containing 18000 equally divided samples.

6.3 Classifier Results

This section gives the results of the tests conducted on the different classifiers as discussed in Chapter 4. The comprehensive set results for all the classifiers investigated can be seen in Appendix D.

6.3.1 Classifier Accuracy

Figure 6.3 shows the classifier accuracies of the different classifiers investigated, for the three sensor cases. The 95% confidence intervals for the accuracies shown were within a 0.2% interval.

When using both anemometric sensors and inertial sensors, most of the classifiers provided classification accuracies greater than 97%, with the bagging classifier providing the best accuracy of 99.45%, and Nearest Neighbour providing the worst accuracy of 74.74%. Decision Trees were the top performing non-ensemble classifier with an accuracy of 99.25%.

The AdaBoost, decision tree, bagging and multi-classifier system trained on anemometric and inertial sensor data had overlapping accuracy intervals. McNemar's test

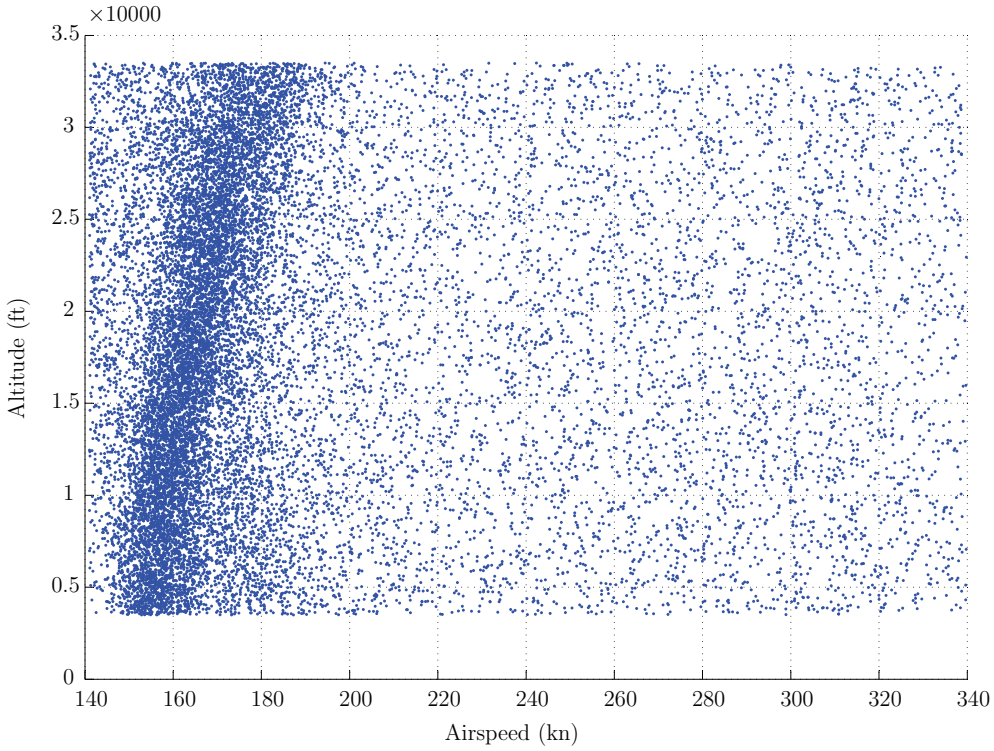

Figure 6.2: Underspeed training data sample distribution

showed that the accuracies produced by these classifiers were statistically significant, with p-values less than 0.05.

When using inertial sensors only, the classification accuracies of the classifiers dropped by about 15.5% on average. When using inertial sensors only, the bagging classifier provided the best accuracy of 89.64% and the Nearest Neighbour provided the worst accuracy of 65.27%.

The AdaBoost, support vector machine and bagging classifier trained on only inertial sensor data had overlapping accuracy intervals, the accuracies produced by these classifiers were statistically significant, as given by a McNemar test.

It is clear that the accuracies of the classifiers were significantly poorer when only inertial sensors were used to detect underspeed. A third classifier was therefore added that detects underspeed using sensor measurements from the inertial sensors supplemented with estimates of the ground speed and horizontal wind speed that are available from the on-board inertial navigation system. The ground speed (V_{sol}) is the speed of the aircraft in the inertial axis-system, the horizontal wind speed ($Wind$) is supplied in the form of a wind magnitude and a wind head. This is used to calculate the wind components in the North and East direction. The addition of the ground speed and wind speed estimates resulted in a significant improvement in the classifier accuracy compared to when only the inertial sensors were used. The multi-classifier system provided the best accuracy of 96.1% and the Nearest Neighbour provided the worst accuracy of 75.67%

The exclusion of the anemometric sensors resulted in a major reduction of classifier accuracy. This is due to the fact that the direct airspeed measurement is not given to the classifiers, and this loss of information resulted in poor classifier accuracy. This shows that the underspeed upset does not manifest as clearly in the inertial sensor measurements as the high angle of attack upset does. It was therefore decided to include estimates of the ground speed and the horizontal wind velocity to the data that is used for the

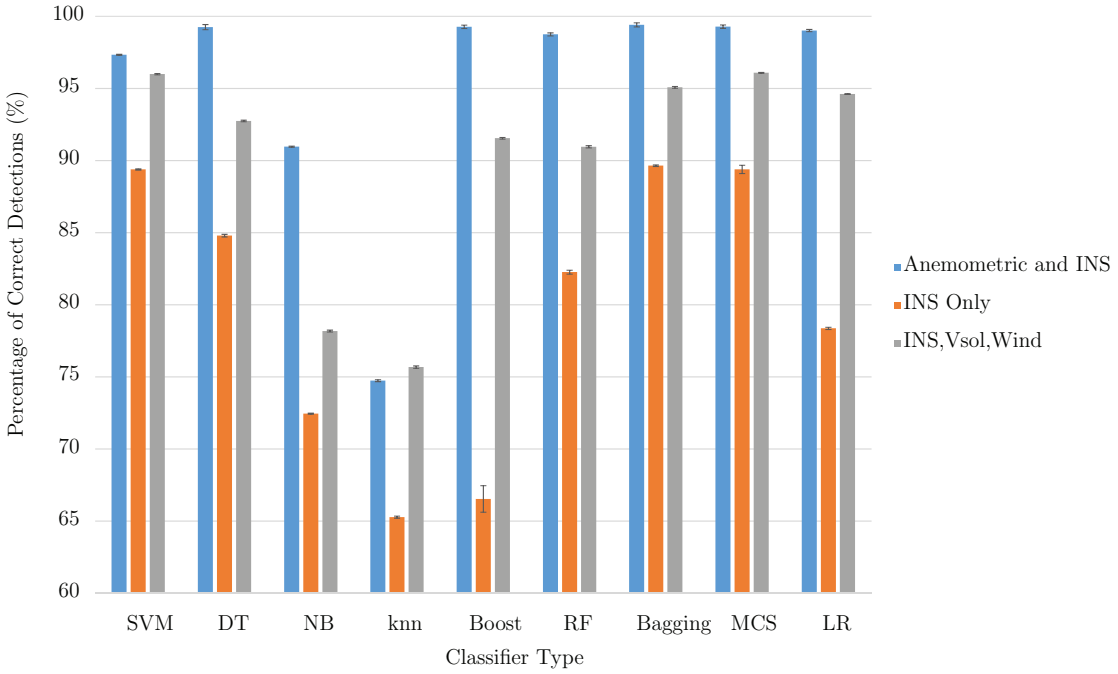

Figure 6.3: The accuracies achieved by the classifiers tested for all three sensor cases

classification. The addition of some airflow information resulted in a large improvement in accuracy. A concern with the addition of these estimates is that the accuracies of the classifiers rely strongly on the accuracy of these estimates. The assumption was made that the estimates have unbiased errors, and were thus modelled by adding zero mean Gaussian noise to the ideal ground and wind speed variables. This might be an optimistic assumption and the classifier accuracies might not be as high as shown. It does show however, that the accuracy of the classifier can be drastically improved with the addition of some airflow information.

6.3.2 Classifier's Sensitivity to Training Configuration Parameters

The default accuracy for the classifiers trained with anemometric and inertial sensors are shown in Figure 6.4, with the blue bars. The median, minimum and maximum accuracies of the random searches are shown with the black error lines. It was found that the accuracy intervals for the support vector machine and the decision tree were larger than 20% and 30% respectively, showing that these non-ensemble classifiers were sensitive to training parameter changes when applied to the problem of underspeed detection. The logistic regression had the smallest accuracy change with an interval of 0.04%, making it extremely insensitive to the training parameters. All the classifiers except the nearest neighbour classifier had an accuracy of the default configuration that was higher than the median.

The default accuracies and the accuracies obtained from the training parameter search when using only the inertial sensors are shown in Figure 6.5. The accuracy intervals

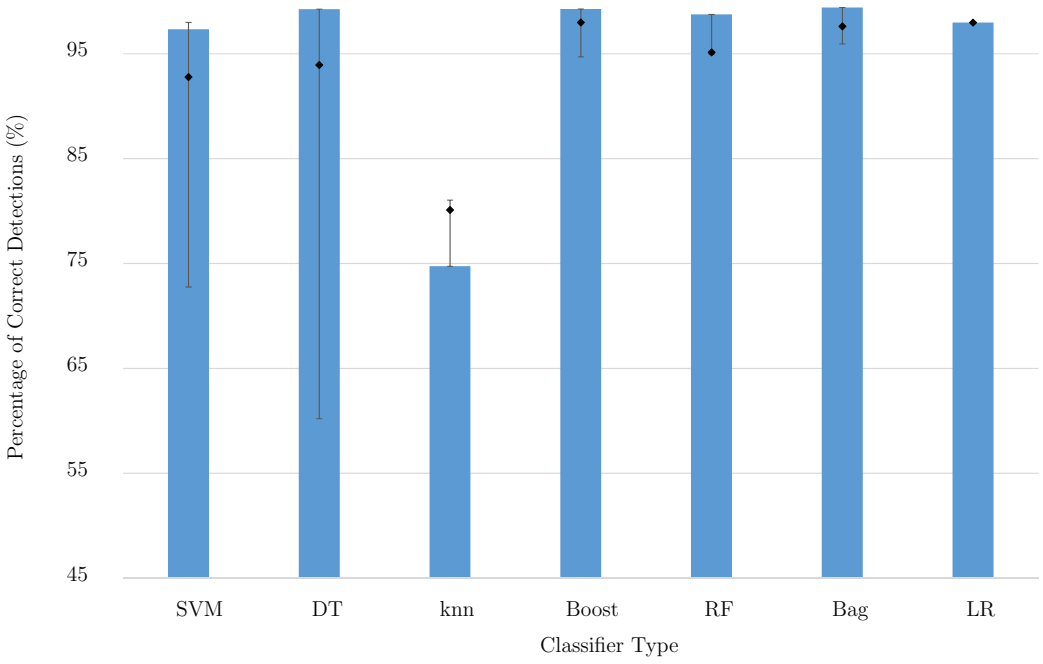

Figure 6.4: Accuracies achieved by a random classifier training parameter search for classifiers trained on anemometric and inertial data.

widened with the exclusion of the anemometric data. The default classifier performances were greater than the median for the majority of the classifiers, with the nearest neighbour and AdaBoost classifiers having a default accuracy smaller than the median. A training configuration parameter change might result in a large improvement for the AdaBoost classifier. The classifiers' accuracies produced by the default configuration were in many cases the best accuracy found by the random search. The default configurations were therefore viable options for classifier evaluation with underspeed upset when using only inertial sensors.

The default classifier accuracies and the minimum, maximum and median accuracies of a random search of classifier training parameters for the classifiers trained on inertial data and estimates of ground and wind speeds, can be seen in Figure 6.6. The majority of the default accuracies were near the maximum accuracy achieved by the search. The nearest neighbour had a default configuration accuracy that was lower than the median. The support vector machine, decision tree and AdaBoost classifiers had accuracy intervals greater than 15%, which shows that these classifiers are sensitive to training parameter changes.

The random search conducted showed that the default classifier configurations were reasonable choices on which to evaluate the accuracies of the different classifiers for underspeed upset detection. The default classifier configurations were therefore used throughout the tests conducted on the classifiers.

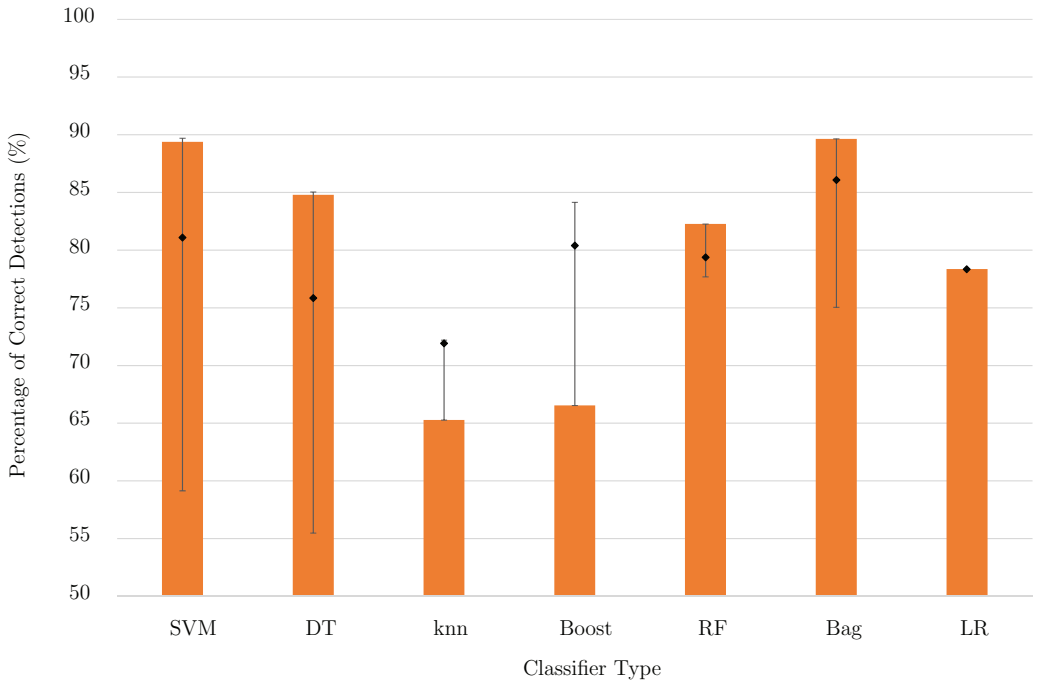

Figure 6.5: Accuracies achieved by a random classifier training parameter search for classifiers trained on only inertial data.

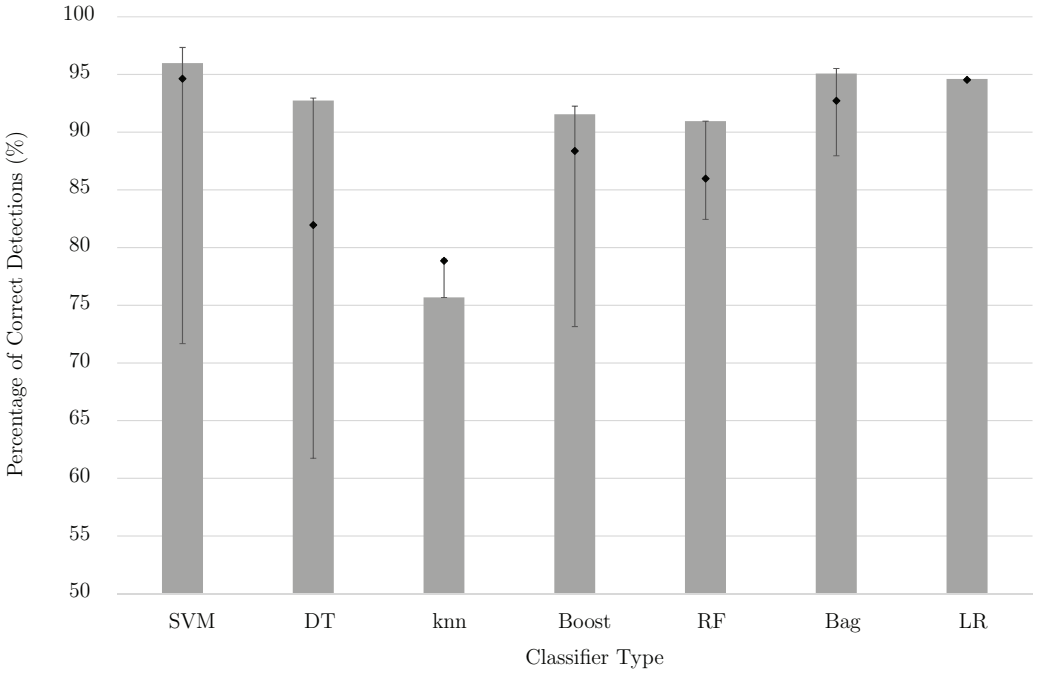

Figure 6.6: Accuracies achieved by a random classifier training parameter search for classifiers trained on inertial data and estimates of ground and wind speeds.

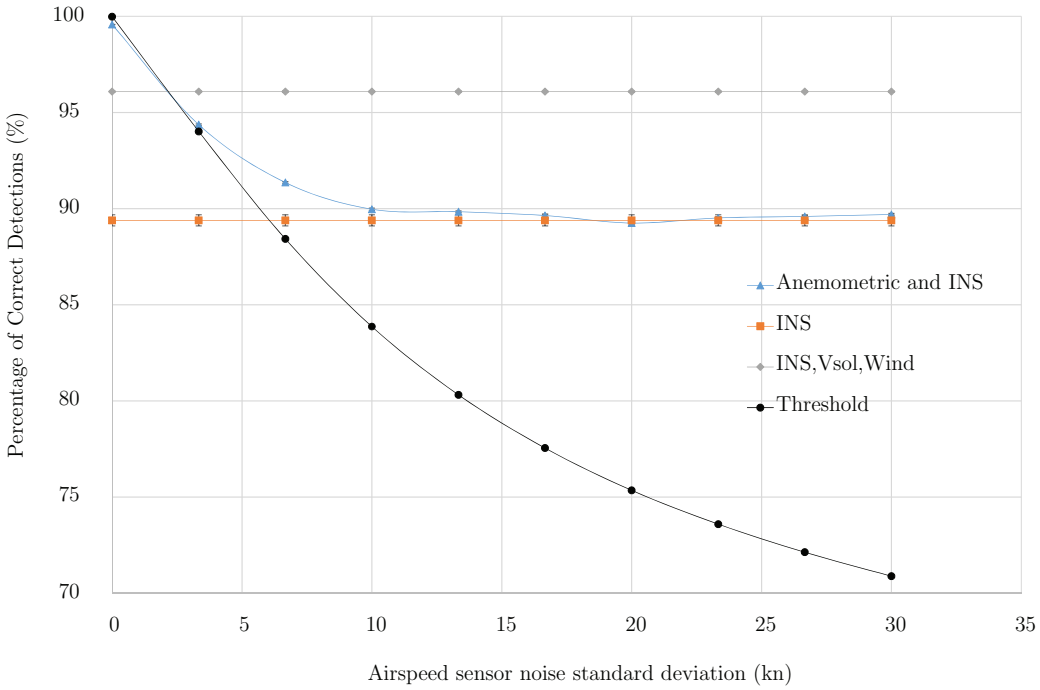

Figure 6.7: Classifier accuracy at different angle of attack sensor noise levels for the MCS

6.3.3 Classification vs. Estimation-based Upset Detection with Noisy Anemometric Sensors

Figure 6.7 shows the classification accuracy of the multi-classifier system for the three sensor cases, compared to the accuracy of simply thresholding a noisy airspeed sensor against the underspeed airspeed. The accuracy when using both anemometric and inertial sensors decreased as the noise increased, and flattened out at 89.5%. The accuracy for sensor cases that excluded the anemometric sensors remained at a constant accuracy. It was found that the classification-based approach outperformed the estimator-based approach when the standard deviation of the airspeed measurement/estimation noise exceeded $3kn$s, as shown in Figure 6.7.

6.3.4 Locations of False Alarms

Figure 6.8 shows the airspeed distribution of the accumulated false alarms (shown in red) and the airspeed distribution for an A330 in normal flight (shown in green). The underspeed distributions shown were around an altitude of $17500ft$. It was found that airspeed separation between the two distribution for the sensor case with the anemometric and inertial sensors available was $\approx 70kn$.

The airspeed separation between the two distributions decreased to $\approx 27kn$ with the exclusion of the anemometric sensor data, as shown in Figure 6.9. This exclusion resulted in a notable increase of false alarms situated around an altitude of $17500ft$, from 3.1% to 14.01%.

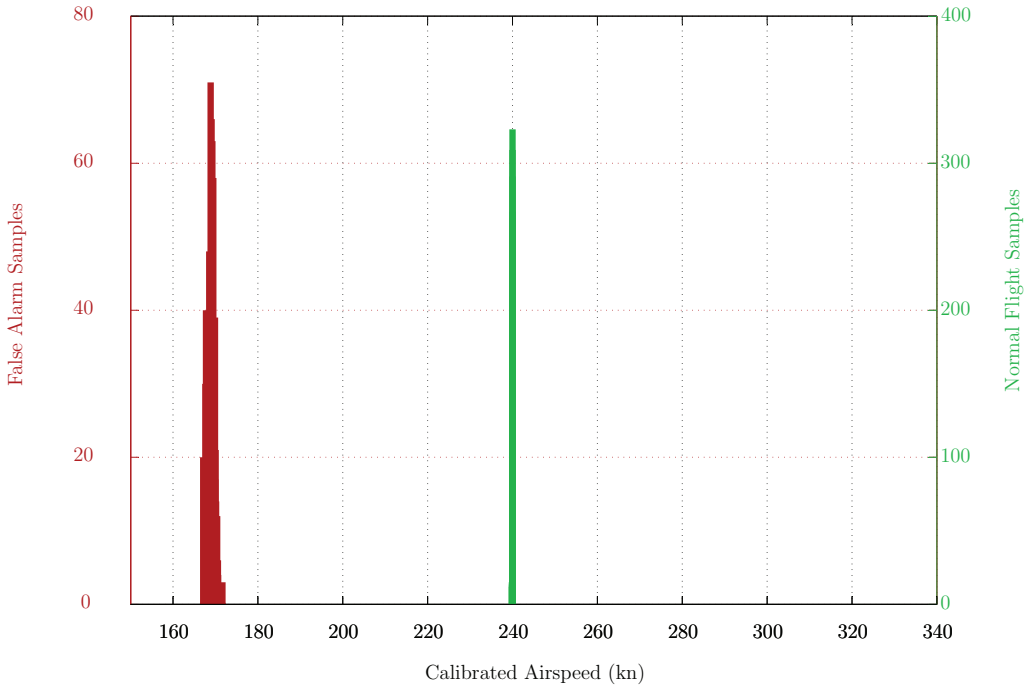

Figure 6.8: The location of the false alarm generated by the MCS trained on anemometric and inertial sensor data.

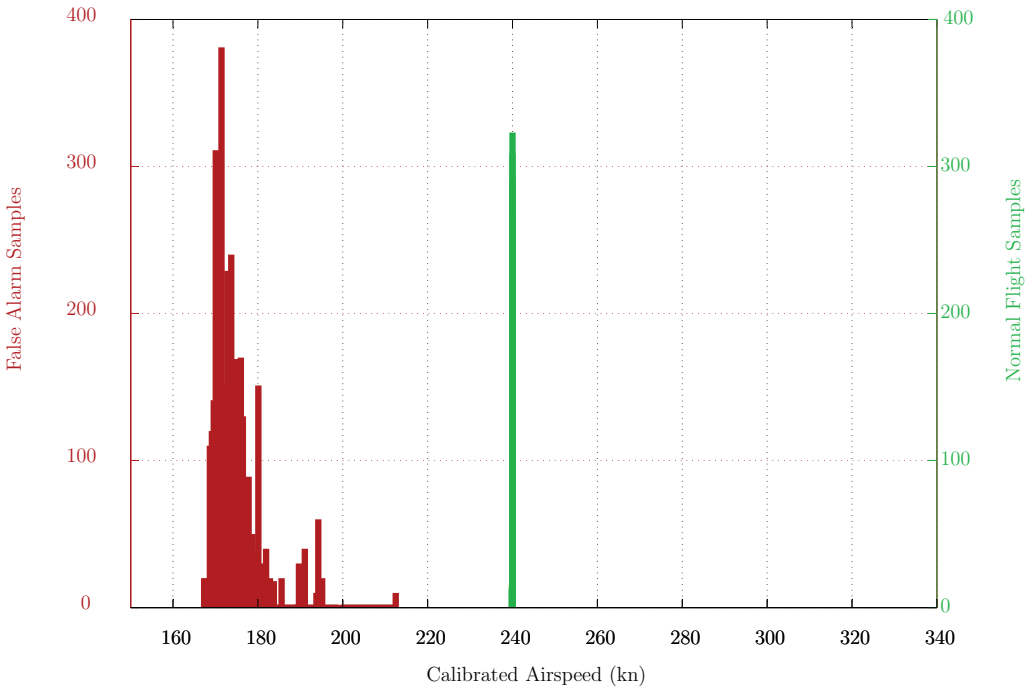

Figure 6.9: The location of the false alarm generated by the MCS trained on only the inertial sensor data.

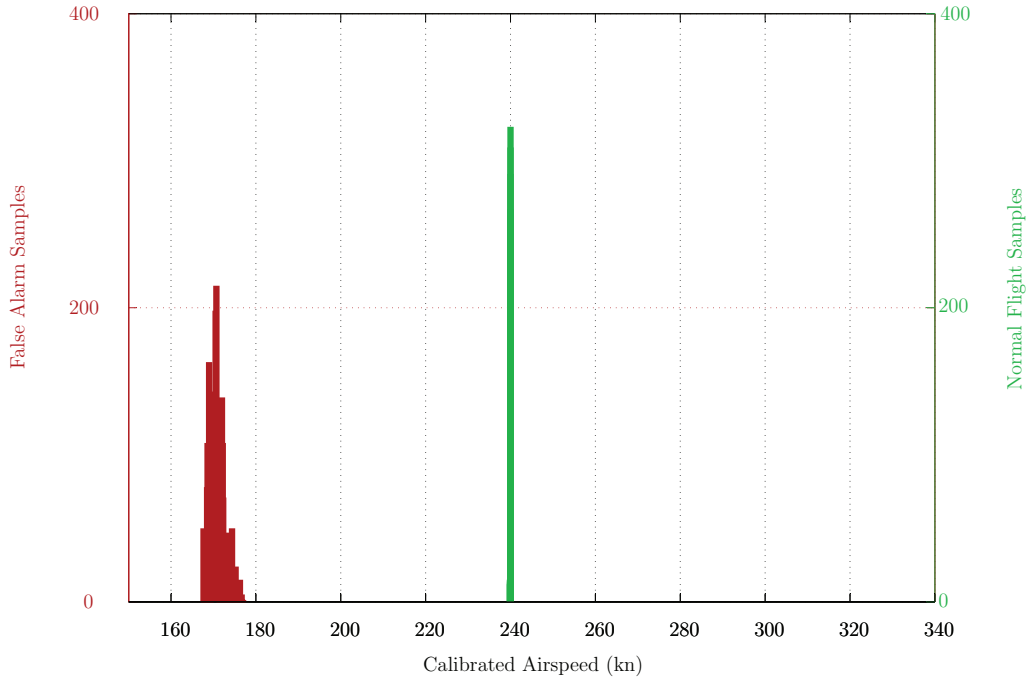

Figure 6.10: The location of the false alarm generated by the MCS trained on inertial sensor data and estimates of the ground and wind speeds.

It was found that airspeed separation between the two distribution for the sensor case with inertial sensors and estimates of ground speed and wind speed was $\approx 65kn$, as shown in Figure 6.10. The underspeed distributions shown were around an altitude of $17500ft$. The percentage of false alarms was 4.07% for the flight point shown.

The airspeed distribution of the false alarms and the airspeed distribution of an A330 in normal flight were sufficiently removed from one another to ensure a reliable upset detection system, for the two sensor case where the anemometric sensors or estimates of the ground speed and wind speed were available. This ensures a reliable upset detection system. When only the inertial sensors were available—without any estimates of ground and wind speed—the false alarms produced by the classifiers were situated near the normal flight domain, this renders it unreliable at low airspeeds within the normal flight domain.

6.4 Simulation of Validation Manoeuvre

Simulations were then performed with the Airbus A330 model to validate the performance of the underspeed upset detection with simulated upset manoeuvres.

The manoeuvres performed to validate the underspeed detection system were chosen so that the aircraft experienced a gradual reduction in airspeed. The rate of airspeed deceleration varied as part of the validation manoeuvre set.

Figure 6.11 shows the false alarms and missed detections generated by the underspeed upset detection system when using both anemometric and inertial sensor data for the given manoeuvre. The false alarms and missed detections generated were in close proximity to the upset boundary, ensuring a reliable underspeed detection system.

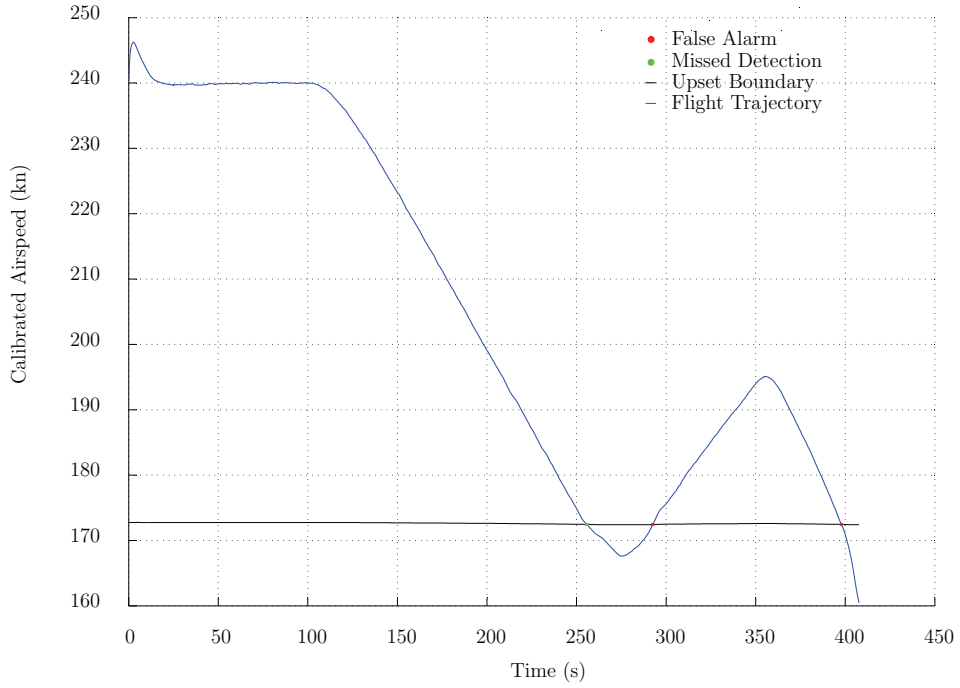

Figure 6.11: False alarms and missed detections generated by the MCS trained on anemometric and inertial data given an underspeed manoeuvre. The speed trajectory is shown in blue

Figure 6.12 shows the false alarms and missed detections for the same flight trajectory but with the underspeed upset detection system using only the inertial sensor data. The number of false alarms and missed detections and the distance from the upset boundary increased significantly. The false alarms that were the furthest away from the upset boundary were at an airspeed of $\approx 195kn$ which is well within the normal flight envelope of the aircraft, and the majority of the underspeed samples were misidentified. This shows that reliable underspeed upset detection with this system is not possible.

Figure 6.13 shows the false alarms and missed detections generated by the underspeed upset detection system when using inertial data supplemented with estimates of the ground and wind speeds obtained from the on-board inertial navigation system. The false alarms and missed detections were low and in close proximity to the underspeed boundary, ensuring a reliable underspeed detection system.

The underspeed upset detection system was validated with flight manoeuvres using a representative simulation of a commercial passenger airliner. The results showed that the false alarms and missed detections lie in vicinity of the upset boundary as found for the random samples in the previous section. This shows that the proposed random training data generation method is an effective way to train classifiers for underspeed upset detection.

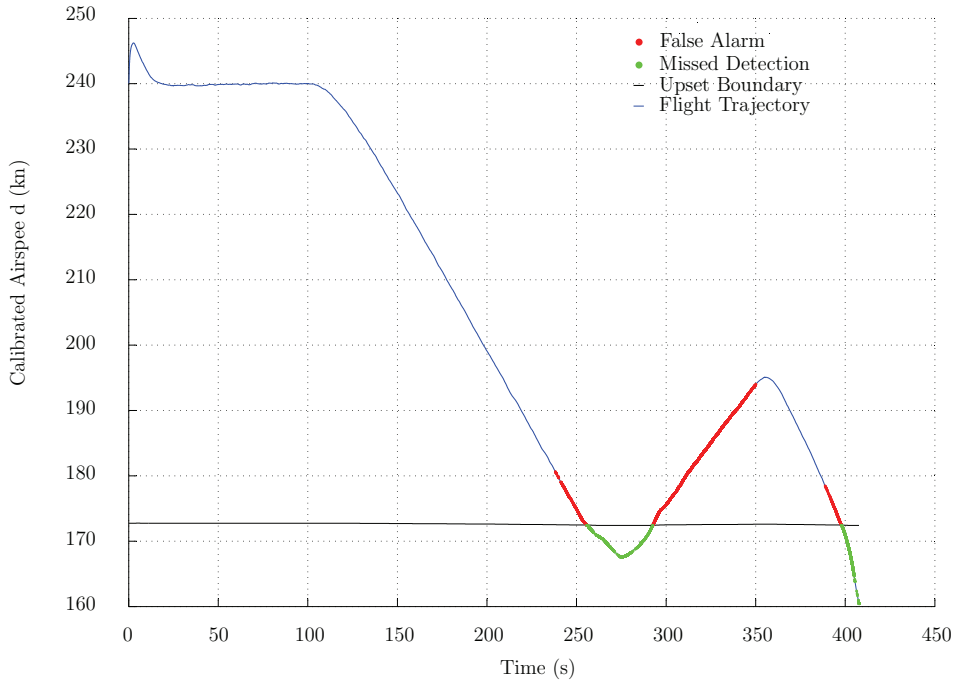

Figure 6.12: False alarms and missed detections generated by the MCS trained on only inertial data given an underspeed manoeuvre. The speed trajectory is shown in blue

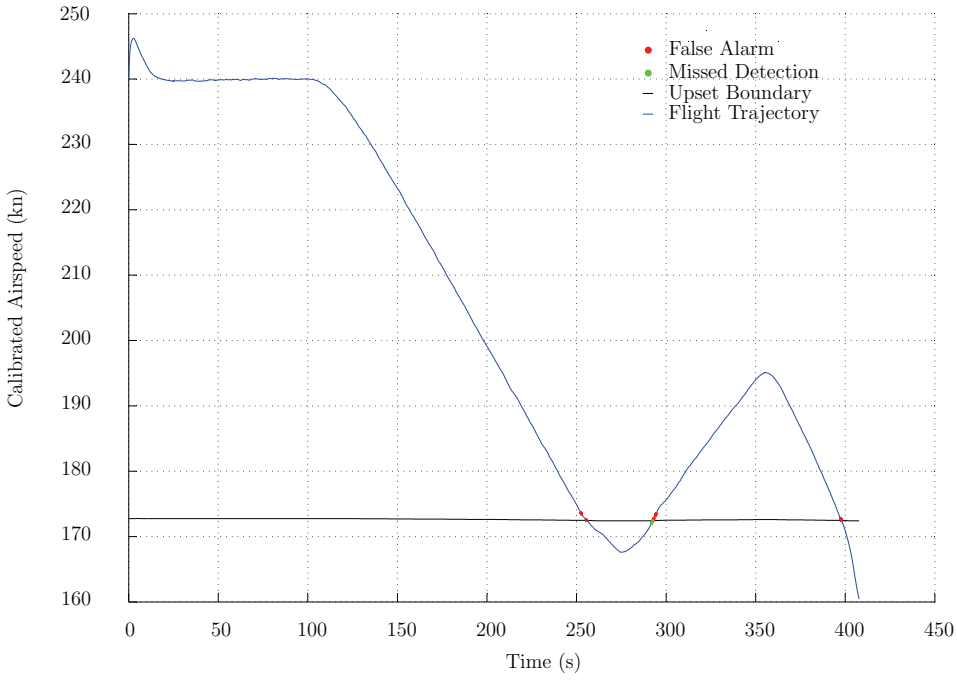

Figure 6.13: False alarms and missed detections generated by the MCS trained on inertial data and estimates of the ground and wind speeds, given an underspeed manoeuvre. The speed trajectory is shown in blue

6.5 Conclusions

The classification-based underspeed upset detection when anemometric and inertial sensors were available, had an exceptionally high accuracy of 99.45% for the bagging classifier, with the false alarms and missed detection in close proximity to the underspeed boundary. This provided a reliable means of underspeed detection using classification techniques. A random search of the classifier training parameters showed that the default classifier configurations were a good basis for classifier evaluation, as the accuracies achieved by the default classifier configuration were among the best accuracies achieved by the random classifier parameter search. The classification-based underspeed detection was compared with the estimation-based approach for underspeed detection, it was found that the classification-based approach outperformed the estimation approach when the standard deviation of the airspeed measurement/estimation noise exceeded $3kn$s. It was found that it was not possible to detect underspeed reliably using classifiers trained only on inertial data, as the false alarms generated were within the normal flight domain. The addition of estimates of ground speed and wind speed supplied sufficient information to accurate and reliably detect underspeed upset. With the multi-classifier system providing an accuracy of 96.1% and with the false alarms and missed detections in close proximity to the underspeed boundary. The multi-classifier system consisted of the three top performing classifiers which were, a support vector machine, bagging classifier and an AdaBoost classifier.

The system was validated with flight manoeuvres using a representative simulation of a commercial passenger airliner.

Chapter 7

Dynamic Pitch Upset Detection

This chapter presents the definition of dynamic pitch and the evaluation of a dynamic pitch upset detection function for commercial passenger airliners that uses classification algorithms operating on a combination of anemometric and inertial sensor measurements.

The chapter starts with the definition of dynamic pitch upset and a description of the classifier training data. The total classifier accuracies and sensitivity to training configuration parameters are shown. A comparison between the classification-based approach and estimation-based approach of upset detection is presented. Next, the locations of the false alarms and missed detections relative to the classification boundary are evaluated and compared to the expected angle of attack and pitch rates encountered in normal flight. Finally, the dynamic pitch detection system is validated with simulated upset manoeuvres using a representative flight dynamics model for a commercial passenger aircraft.

7.1 Dynamic Pitch Upset Definition

Dynamic pitch upset is a predictive form of high angle of attack upset. We define a dynamic pitch upset as a condition where the predicted angle of attack given the current angle of attack and pitch rate will exceed the critical angle of attack even if maximum elevator deflection is immediately applied to oppose the pitch rate. The maximum angle of attack that would be reached if full opposing elevator is applied from a given initial angle of attack and pitch rate is determined via forward simulation. This maximum angle of attack reached is compared to the critical angle of attack used in the high angle of attack upset detection function. The aircraft is considered to be in a dynamic pitch upset if the maximum angle of attack reached exceeds the critical angle of attack. The maximum angle of attack is calculated at different altitudes and airspeeds (flight points). Due to the time scale separation between the flight path angle and the pitch angle, the flight path angle is assumed to remain constant during the manoeuvre. The rate of change of the angle of attack is therefore assumed to be equal to the pitch rate during the manoeuvre.

Figure 7.1 shows the dynamic pitch upset boundary at a specific flight point (altitude and airspeed). It was defined that the minimum angle of attack, at negative pitch rates, may not exceed an angle of attack less than zero, as shown in the blue line in Figure 7.1. The boundary was calculated at different altitudes and airspeeds to form a four-dimensional boundary over pitch rate (Q), angle of attack, altitude and airspeed.

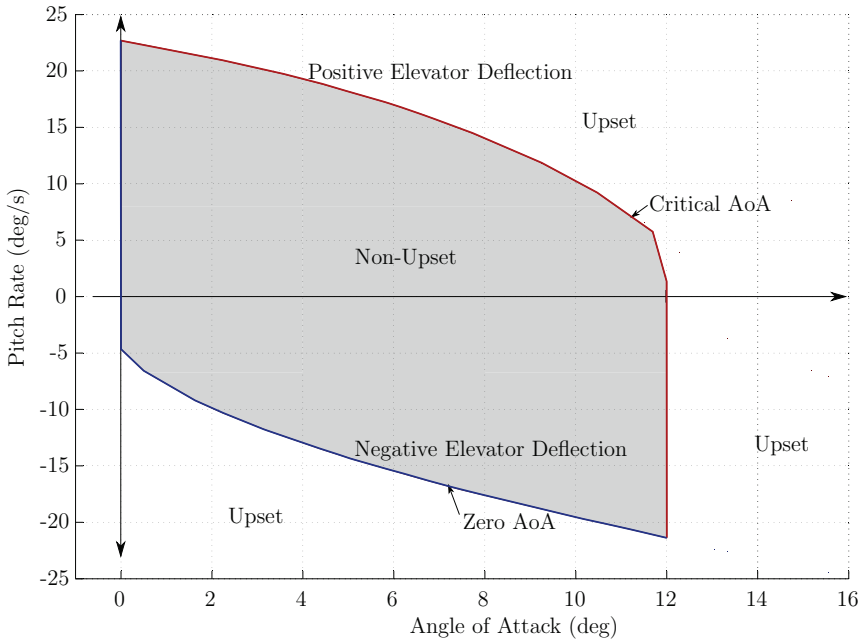

Figure 7.1: The dynamic pitch upset boundary for an Airbus A330 aircraft. The upper angle of attack limit is shown in red and the lower angle of attack limit is shown in blue.

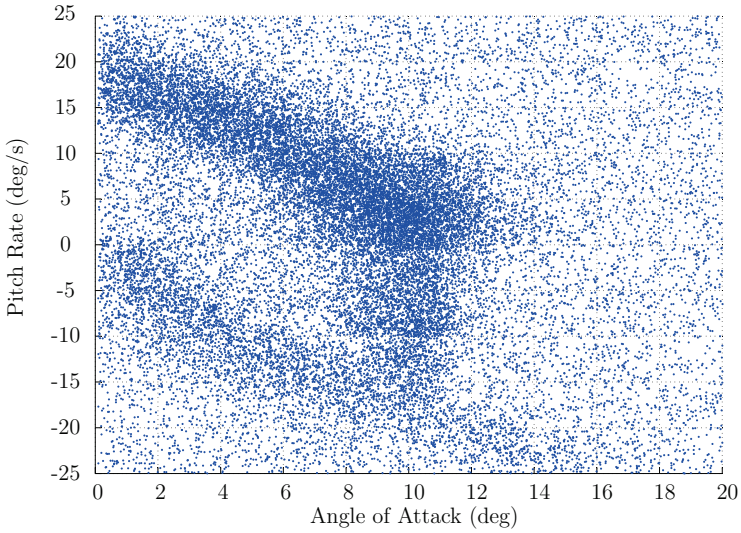

Figure 7.2: The training data set's sample distribution for dynamic pitch upset.

7.2 Training Data

The data set that was used for the classifier training and testing was generated from the Airbus A330 simulation model.

Figure 7.2 shows the pitch rate and angle of attack sample distribution of the training data set. The tight normal distributed samples around the dynamic pitch upset boundary had an angle of attack standard deviation of 0.5° and a pitch rate standard deviation of 1°. A description of the distributions and their associated ranges used to generate the training data are shown in Appendix A.

7.3 Classifier Results

A number of different classifiers were trained on the data set to detect dynamic pitch upset.The training procedure as discussed in Chapter 4, was followed.

This section gives the results of the tests conducted on the different classifiers, as discussed in Chapter 4. The comprehensive set results for all the classifiers investigated can be seen in Appendix E.

7.3.1 Classifier Accuracy

The different classifiers were then tested on the testing data set and their performances as classifiers were evaluated, as discussed in Chapter 4.

Figure 7.3 shows the classifier accuracies of the different classifiers investigated, for both of the sensor cases. The 95% confidence intervals for the accuracies shown were within a 0.15% interval.

When using both anemometric sensors and inertial sensors the bagging classifier provided the best accuracy of 93.9%, and logistic regression classifier provided the worst accuracies of 68.3%. Decision Trees were the top performing non-ensemble classifier with an accuracy of 91.7%.

The Naïve Bayes and nearest neighbour classifier trained on anemometric and inertial sensor data were the only classifier with overlapping accuracy intervals. McNemar's test showed that the accuracies produced by the Naïve Bayes and nearest neighbour classifiers were statistically significant with p-values less that 0.05.

When using inertial sensors only, the classification accuracies of the classifiers dropped by about 4.05% on average. When using inertial sensors only, the multi-classifier system containing a support vector machine, bagging classifier and an AdaBoost classifier, provided the best accuracy of 91.4% and the logistic regression classifier provided the worst accuracy of 66.13%.

The accuracies were considerably lower than the classifier accuracies for the high angle of attack and underspeed upset detection systems. This can be attributed to the complexity of the four dimensional dynamic pitch upset boundary.

7.3.2 Classifier's Sensitivity to Training Configuration Parameters

An investigation was performed to explore the sensitivity of the classifier accuracies to the training parameters that were used, as discussed in Chapter 4.

The default accuracy for the classifiers trained with anemometric and inertial sensors are shown in Figure 7.4, with the blue bars. The median, minimum and maximum accuracies of the random searches are shown with the black error lines.

It was found that the classifiers' searched parameter space produced a large accuracy interval. This shows that a small change in the classifier's training parameter might result in a large accuracy change, and therefore requires a great amount of human input to optimise.The nearest neighbour classifier was the only classifier where the default classifier training configuration produced the lowest accuracy. This shows that the nearest neighbour might greatly benefit from some training parameter optimisation. The rest of the classifiers' default accuracies were greater than the medians of the accuracies achieved with the parameter search. This shows that the default configurations were reasonable choices on which to evaluate the accuracies of the different classifiers for high angle of

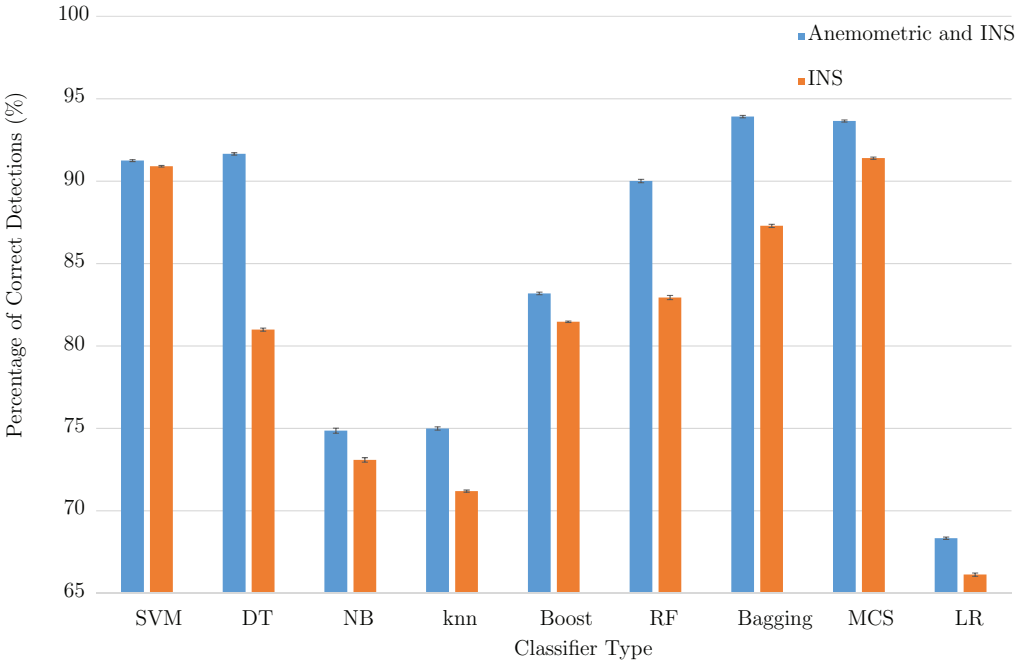

Figure 7.3: The percentage of correct detections made by the different classifiers for the dynamic pitch upset class.

attack upset detection. The logistic regression classifier showed little change in accuracy with different training parameters.

The default accuracies and the accuracies obtained from the training parameter search for only the inertial sensors available are shown in Figure 5.7. The variation in the classifier accuracies achieved by the random training parameter search showed little change with the exclusion of the anemometric data. The classifiers' accuracies produced by the default configuration were in many cases the maximum accuracy found by the random search. The default configurations were therefore viable options for classifier evaluation with high angle of attack upset using only inertial sensors in mind.

The random search conducted showed that the default classifier configurations were reasonable choices on which to evaluate the accuracies of the different classifiers for dynamic pitch upset detection. The default classifier configurations were therefore used throughout the tests conducted on the classifiers.

7.4 Classification vs. Estimation-based Upset Detection with Noisy Anemometric Sensors

The classification accuracies of the different classifiers were compared to the accuracy of performing upset detection by simply thresholding direct measurements or estimates of the angle of attack and pitch rate against the angle of attack and pitch rate boundary of the dynamic pitch upset definition, as discussed in Chapter 4.

Figure7.6 shows the classification accuracy of the multi-classifier system compared to

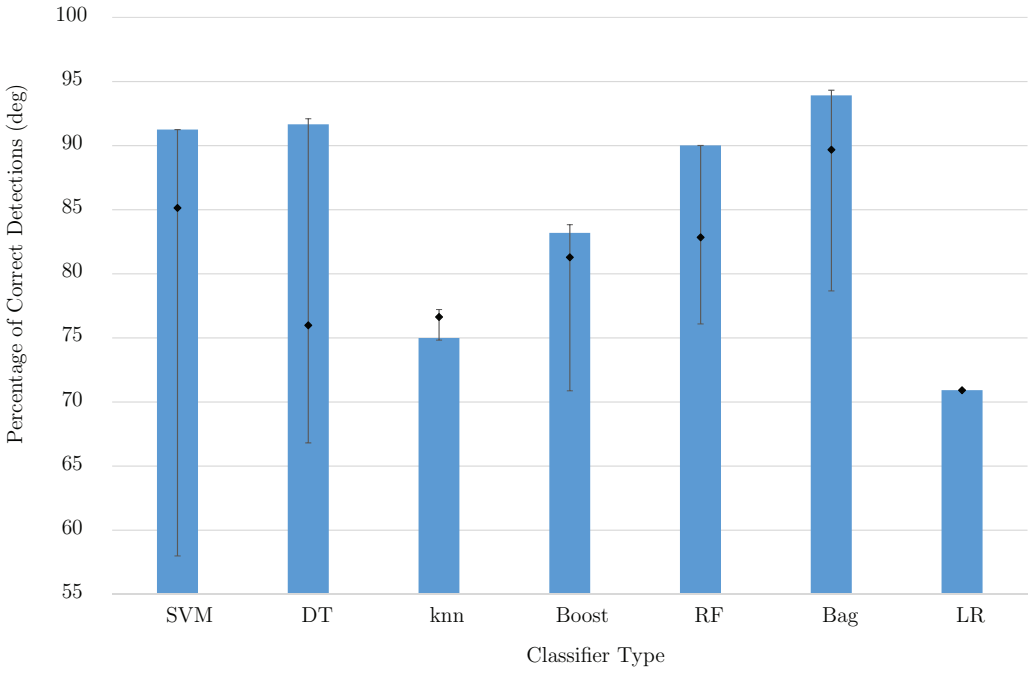

Figure 7.4: Accuracies achieved by a random search for the training parameters of the classifiers trained on anemometric and inertial sensor data.

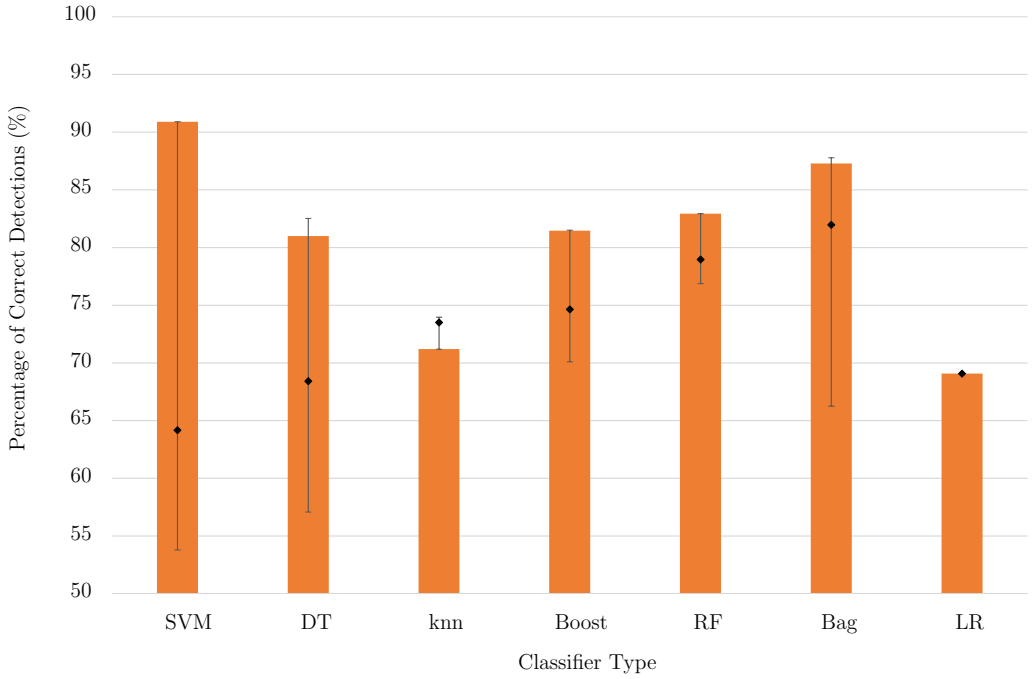

Figure 7.5: Accuracies achieved by a random search for the training parameters of the classifiers trained on only inertial sensor data.

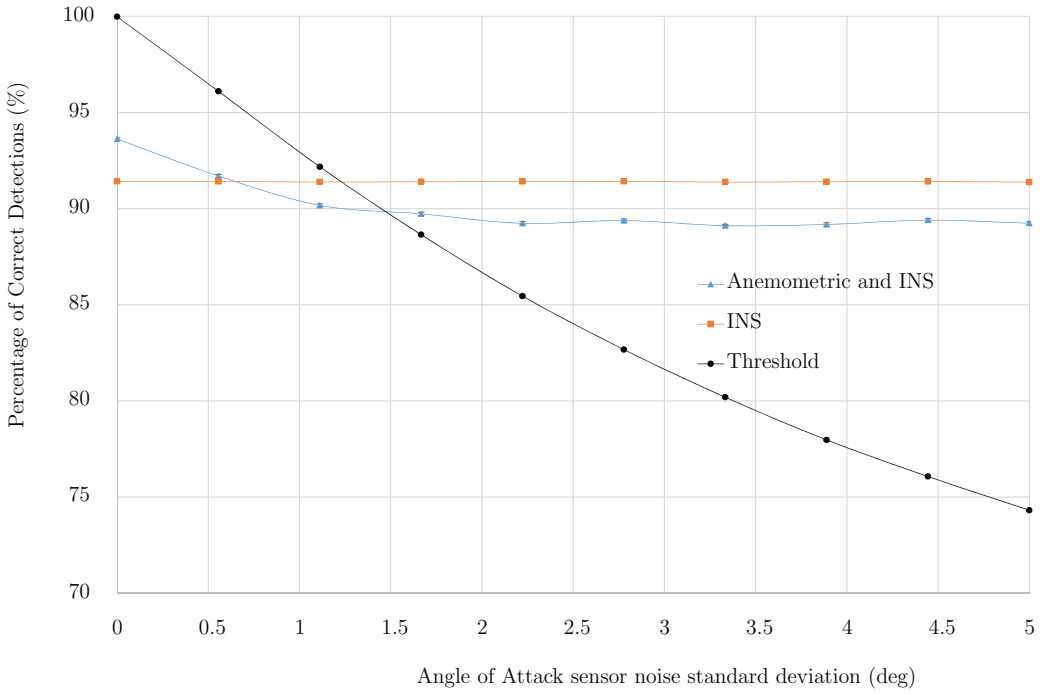

Figure 7.6: The multi-classifier system's accuracy at different angle of attack sensor noise intensities for both sensor cases. The percentage of correct detections made by the threshold method is shown in black.

the accuracy of simply thresholding a noisy angle of attack against the angle of attack of the dynamic pitch upset boundary. The classification accuracy when using both anemometric and inertial sensors flattened out at 89%, as the angle of attach noise increased. The accuracy when only using inertial sensors remained constant at 91%. It was found that the classification-based approach outperformed the estimator-based approach when the standard deviation of the angle of attack measurement/estimation noise exceeded 1.2 degrees, as shown in Figure 7.6.

7.4.1 Locations of False Alarms

The practical performance of the different classifiers as dynamic pitch upset detection functions were evaluated, as discussed in Chapter 4.

Figure 7.7 shows the false alarms generated by the multi-classifier system in the $Q-\alpha$ plane. The normal flight sample under moderate turbulence is shown in green, with the upset boundaries shown in black. The normal flight samples were generated at a flight point of $17500ft$ and $240kn$. All the upset boundaries within the flight point interval are shown as black lines in the figure. The false alarms generated by the multi-classifier system trained on anemometric and inertial sensor data and the false alarms generated by the dynamic pitch upset detection system trained on only the inertial data are shown in Figures 7.7 and 7.8 respectively. The dynamic pitch upset detection system trained on anemometric and inertial sensor data generated false alarms in close proximity to the dynamic pitch upset boundary, with no false alarms in the normal flight domain, with the majority of the false alarms at low pitch rates and high angle of attack values. The

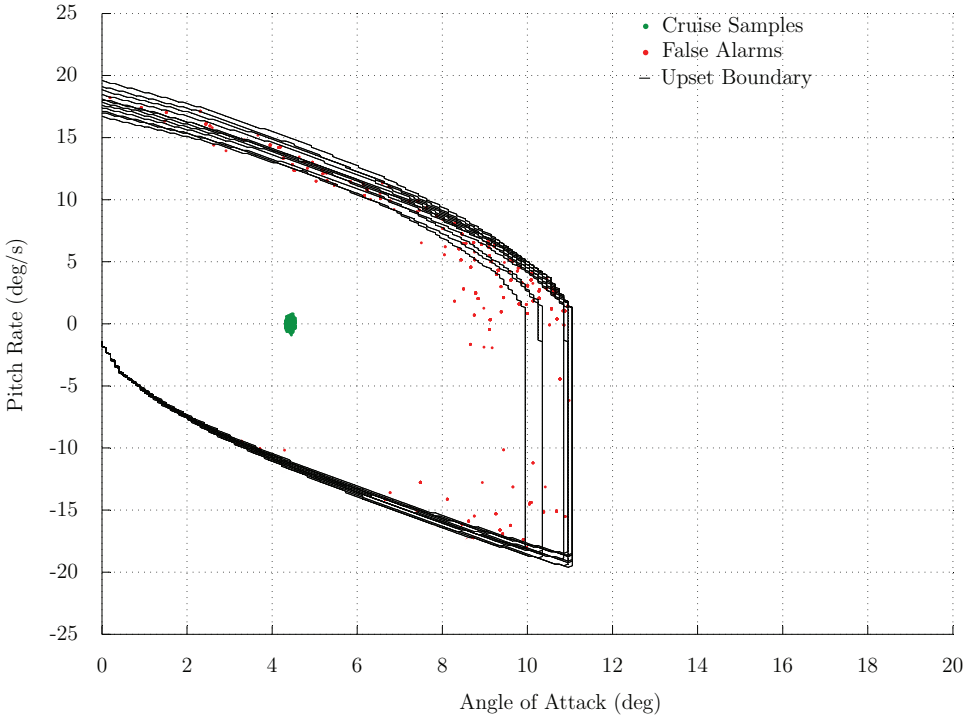

Figure 7.7: The positions of the false alarms generated by the MCS trained on anemometric and inertial sensor data. The sample distribution of straight and level flight in moderate turbulence is shown in green.

number of false alarms generated by the dynamic pitch upset detection system trained on only the inertial data increased. The distance of the samples from the upset boundary increased with the exclusion of the anemometric sensors. However, some false alarms did occur within the normal flight domain.

The false alarms and normal flight samples of an A330 in normal flight were sufficiently removed from one another to ensure a reliable upset detection system, for the case where both the anemometric and inertial sensors were available. The exclusion of the anemometric sensors resulted in the occasional false alarm within the normal flight domain. These false alarms can easily be removed with a low pass filter. Since the sporadic nature of the false alarms, it will be a single sample within a time sequence that can easily removed with filtering.

7.5 Simulation of Validation Manoeuvre Results

Simulations were then performed with the Airbus A330 model to validate the performance of the dynamic pitch upset detection with simulated upset manoeuvres.

The elevator was oscillated between the minimum and maximum allowable deflections in order to generate the manoeuvre shown in Figures 7.9 and 7.10. The manoeuvre shown was completed at a flight point of $17500ft$ and $240kn$.

Figure 7.9 shows the flight trajectory of a manoeuvre performed; the arrows indicate the direction of the flight trajectory. The false alarms and missed detections generated by the dynamic pitch upset detection system trained on anemometric and inertial sensor

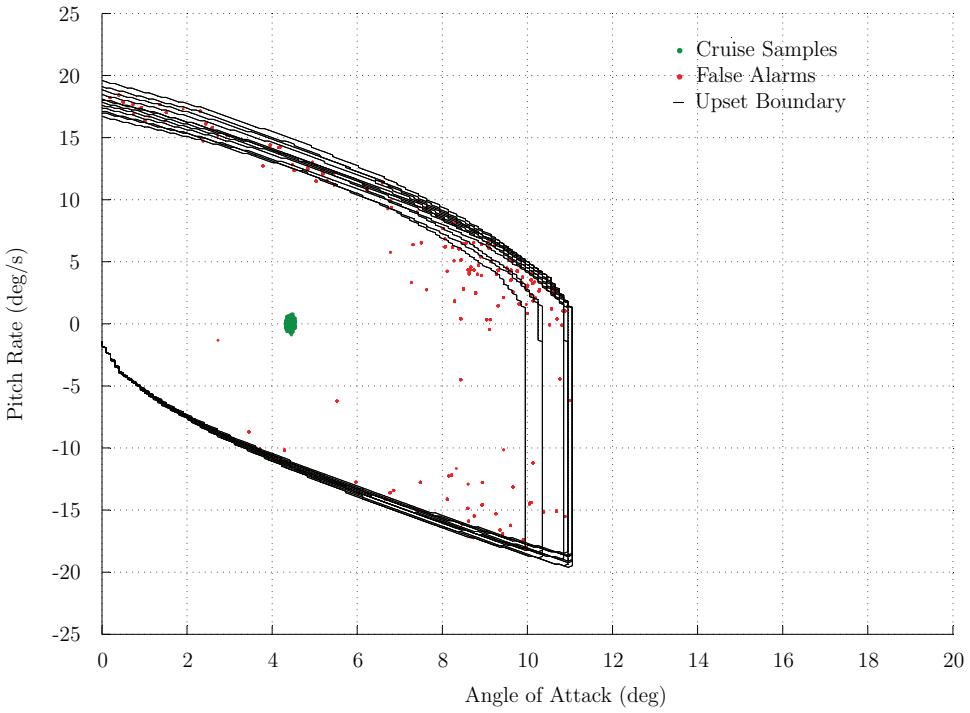

Figure 7.8: The positions of the false alarms generated by the MCS trained on only inertial sensor data. The sample distribution of straight and level flight in moderate turbulence is shown in green.

data are shown. The false alarms and missed detections were in close proximity to the upset boundary, ensuring reliable dynamic pitch upset detection.

The false alarms and missed detections generated by the dynamic pitch upset detection system trained on only inertial sensor data can be seen in Figure 7.10. This sensor case generated considerably more false alarms than the case where both the anemometric and inertial sensor data were available. False alarms stretched deep within the normal flight domain, which is cause for concern. The missed detections remained in close proximity to the upset boundary and did not increase considerably.

The validation manoeuvres provided valuable information about shortcomings in the classifier training data, that were corrected in the results shown. The dynamic pitch upset detection system was validated using flight manoeuvres representative of a real-world scenario. The results showed that the false alarms and missed detections lie in vicinity of the upset boundary as found for the random samples in the previous section. This shows that the proposed random training data generation method is an effective way to train classifiers for dynamic pitch upset detection, for the case where the anemometric and inertial sensors available. The upset detection system trained on only inertial sensors produced false alarms within the normal flight domain. A study should be conducted to determine whether these false alarms can be mitigated.

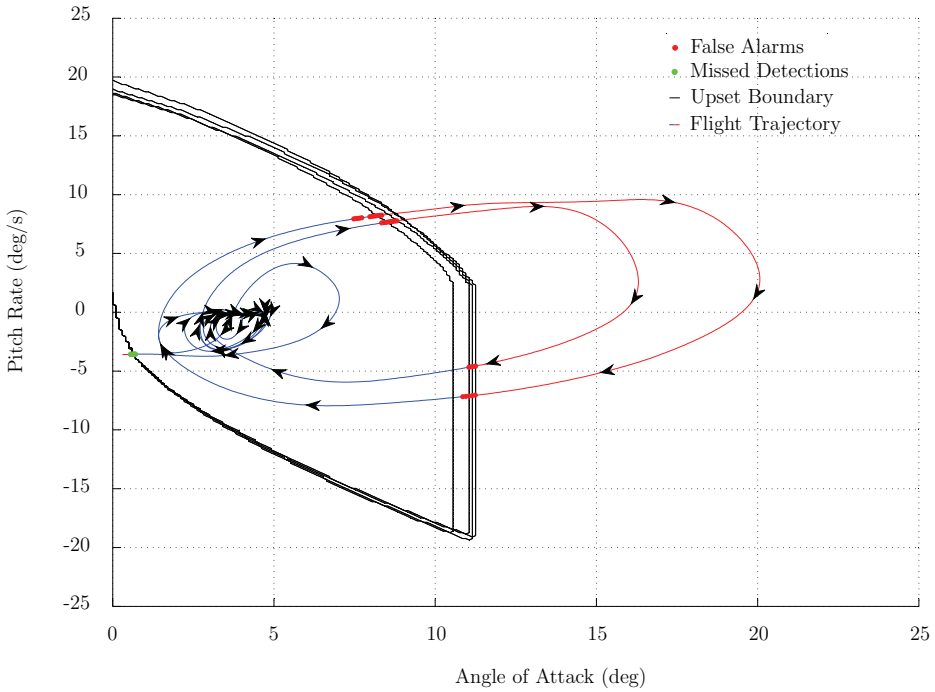

Figure 7.9: False alarms and missed detections generated by the multi-classifier system trained on anemometric and inertial sensor data during an upset manoeuvre.

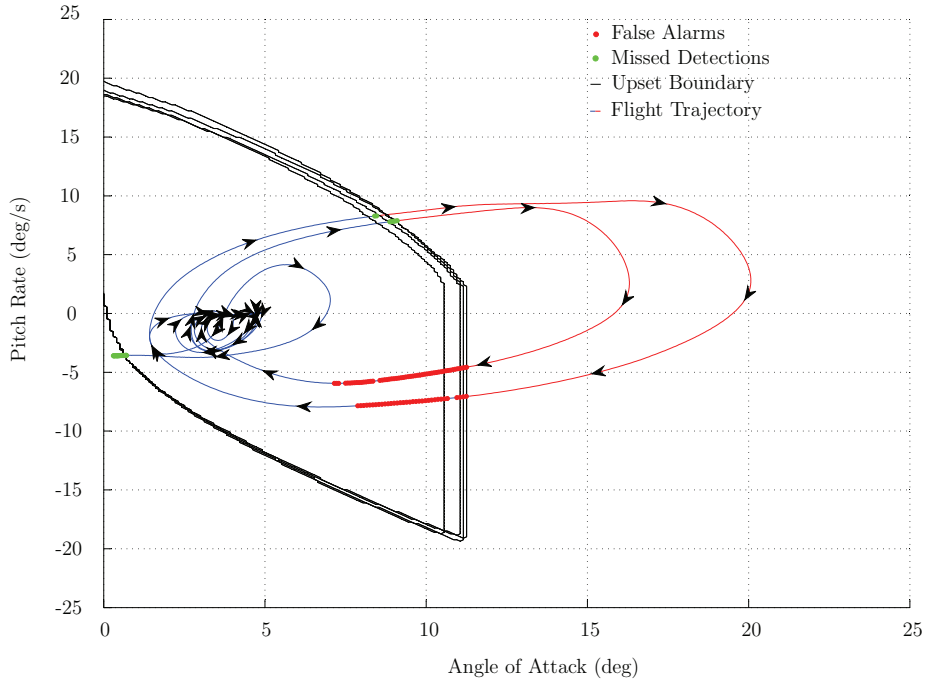

Figure 7.10: False alarms and missed detections generated by the multi-classifier system trained on only inertial sensor data during an upset manoeuvre.

7.5.1 Conclusions

The bagging classifier using both anemometric and inertial sensor data were the top performing classifier with an accuracy 93.9$ with the false alarms and missed detections in close proximity ot the upset boundary, allowing reliable dynamic pitch upset detection. The classification-based approach for dynamic pitch outperformed the estimation approach for estimation errors on the angle of attack greater than 1.2 degrees. The exclusion of the anemometric sensor data resulted in a 2% drop in classifier accuracy with the multi-classifier system providing the best accuracy of 91.4%. The dynamic pitch upset detection system generated sporadic false alarms in the normal flight domain.

The system was validated with flight manoeuvres using a representative simulation of a commercial passenger airliner. When using anemometric and inertial sensor data, the system performed as expected with the false alarms in close proximity to the upset boundary. When using only the inertial sensor data, the dynamic pitch upset detection system generated false alarms within the normal flight domain. This may be a result of the upset boundary being to complex for the classifiers to train on, and requires further investigation.

Classification algorithms proved to be a viable option for dynamic pitch upset detection. The multi-classifier system consisting of a support vector machine, bagging classifier and an AdaBoost classifier had the best overall performance for dynamic pitch upset detection, with the false alarms in close vicinity of the upset boundary and achieving reliable upset detection during the validation manoeuvres. The upset detection system when using only inertial sensors produced sporadic false alarms within the normal flight domain. A study should be conducted to determine whether these false alarms can be mitigated, with the use of a simplified upset boundary or with filtering.

Chapter 8

Conclusions and Recommendations

This chapter consists of a summary of the work done in this study, concluding remarks on passenger aircraft upset detection using classification techniques, and recommendations for future work on upset detection.

8.1 Summary and Conclusions

In this study, three aerodynamic upset detection function using classification algorithms were developed and tested on an Airbus A330 model. The upset detection systems made use of anemometric and inertial sensor data in typical conditions. A secondary system was developed that only uses data outputs from the inertial navigation unit to detect upset in case of the loss of any or all of the anemometric sensors. It was assumed that a fault detection and isolation system would be in place which could detect a faulty sensor; this would be used to switch between the two upset detection systems. The system developed in this study will enable pilots of commercial passenger aircraft to detect and recognise upset events in order for the correct recovery actions to be taken.

The three upsets investigated were high angle of attack upset, underspeed upset, and a predictive form of high angle of attack upset named dynamic pitch upset. An aircraft is considered to be in a high angle of attack upset when the aircraft's angle of attack exceeds the angle of attack at which the maximum lift is produced (the critical angle of attack). Underspeed was defined as the airspeed that results in a trim angle of attack that is higher than the critical angle of attack, resulting in the inability to fly at that airspeed without stalling. Dynamic pitch predicts high angle of attack upset given a pitch rate and angle of attack, when a maximum elevator deflection is applied to counter the pitch rate. The maximum angle of attack that would be reached was determined with a forward simulation and compared with the critical angle of attack.

A wide variety of classifiers were investigated in order to determine the feasibility of classification for upset detection. The classification algorithms investigated can be divided into two categories: non-ensemble and ensemble. The non-ensemble classification algorithms investigated for upset detection were support vector machine, decision tree, naïve Bayes, nearest neighbour and logistic regression. These classifiers all make use of different assumptions when classifying a sample. The ensemble classifiers investigated were AdaBoost classifier, random forest, bagging and a multi-classifier system.

Training the classifiers on simulated upsets was not feasible due to the small number of upset samples generated compared to the non-upset samples, and for optimal classifier training, equal numbers of upset and non-upset samples are required. It was also not

practical to recreate all possible upset events. A novel sampling strategy was therefore devised to train the classifiers. The training data set comprised of the union between two separately generated data sets. The first data set was widespread, sampled over the entire aircraft state space. This was to ensure that the classifiers were able to classify any possible aircraft state combination, even if it was not normally reachable using flight control inputs. The second data set consisted of a tight normal distribution around the upset boundaries. This was to add resolution around the upset boundaries so that the classifiers could accurately classify samples that were near the upset boundaries.

The samples generated from the sensor measurements were labelled using the upset definitions as described, and then used to train the classifiers. Learning curves were used to ensure that all the classifiers were sufficiently trained, making for an even comparison between the classifiers. The classifiers were set up to perform at or near an equal error rate using ROC curves. Tenfold Monte Carlo cross-validation was used to reduce any training data bias, and training of all the non-deterministic classifiers was repeated ten times to reduce training bias. These repetitions were used to calculate a 95% confidence interval on all the results.

The support vector machine and logistic regression were the most accurate of the non-ensemble classifiers, and the naïve Bayes and nearest neighbour classifiers were the least accurate. The ensemble classifiers that had the highest accuracies were the multi-classifier system and the bagging classifier, and the random forest was the least accurate. The multi-classifier system generalised the best, with little over-fitting. The classifier trained on data from the inertial navigation system proved to be robust to anemometric sensor noise, adding redundancy to the anemometric sensors. The missed detections and false alarms generated by the multi-classifier system only occurred close to the classification boundaries and there were neither missed detections that were deep into the upset domain, nor false alarms that were well within the domain of normal flight. Manoeuvres that entered and exited the upset domain were used to test the developed upset detection systems with a real-world scenario. A random search within the classifier training parameters space showed that the default classifier configurations produced relatively high accuracies, making it a viable basis of classifier comparison. The search also showed the sensitivity of the different classifiers to their training configurations, with the support vector machine and the decision tree being the most sensitive and the logistic regression classifier the least sensitive.

The high angle of attack upset detection system was able to detect the upset with 98.8% accuracy, when anemometric sensors were available. The false alarms were in close proximity of the upset boundary. When only inertial sensors were used, the detection accuracy of the system was reduced to 92.2%, with all the false alarms and missed detections still in the vicinity of the upset boundary. The validation manoeuvres corroborated the results shown, proving that the sample-based approach for the training of the classifiers was valid.

The underspeed upset detection using both anemometric and inertial sensor measurements was able to detect the underspeed with 99.3% accuracy when anemometric sensors were available, with the false alarms within 5 knots of the underspeed boundary. When only inertial sensors were used, the underspeed detection accuracy reduced by 10% to an accuracy of 89.4%. The false alarms occurred further away from the underspeed boundary resulting in false alarms within the normal flight domain. The underspeed detection function using only inertial sensor data was then augmented by adding estimates of the ground speed and wind speed (which are also available from the inertial navigation system on typical passenger aircraft) to the sensor data used to perform the underspeed classi-

fication. The addition of these estimates resulted in a significant accuracy improvement to 96.1%, with the false alarms returning to the vicinity of the underspeed boundary.

The complexity of the dynamic pitch upset boundary resulted in false alarms being not as close to the dynamic pitch boundary as desired. The upset detection system had an accuracy of 93.6% when both anemometric and inertial sensor data was available, with the false alarms in close vicinity of the upset boundary. When only inertial sensors were used, the detection accuracy was reduced by 2.2% to 91.4%. The false alarms remained in close proximity to the upset boundary. However, some false alarms did occur in the normal flight domain.

The classification-based upset detection system developed provided a reliable method for detecting high angle of attack, underspeed and dynamic pitch upsets, provided that both the anemometric and inertial sensors are available. The false alarms and missed detections generated by the system were in close proximity of the upset boundaries, resulting in an exceptionally low probability of false alarms being generated during normal flight, contributing to the reliability of the system. The high angle of attack upset detection system was the only upset detection function that maintained reliable upset detection, when only the inertial sensors are available. The false alarms and missed detections were in close proximity of the high angle of attack boundary resulting in an accurate and reliable system. Estimates of wind and ground speed were added to the underspeed upset detection system functioning on only the inertial data, to preserve reliable underspeed upset detection. The dynamic pitch upset detection system operating with only inertial sensors, produced false alarms within the normal flight domain. This may cause pilot distrust, rendering the dynamic pitch upset detection system unreliable. The dynamic pitch system is a predictive form of high angle of attack upset detection. The loss of anemometric sensors therefore only inhibits the ability to predict high angle of attack upset, and not the ability to detect high angle of attack upset. It is proposed to split the dynamic pitch upset boundary into two domains, namely pitch rates grater than zero and pitch rates less than zero. The classifiers trained on this simplified version of the upset boundary may provide a reliable method for dynamic pitch upset detection.

The validation manoeuvres initially provided valuable information regarding shortcomings in the classifier training data. It was found that the accuracy of the classifiers are greatly effected by the training data, and that special care should be given to the features ranges used to train the classifier. The validation manoeuvres showed inadequate feature ranges that were corrected. The validation manoeuvres lastly showed that the false alarms and missed detections are in close proximity of the upset boundary, thus validation the system.

The multi-classifier system provided the most accurate and reliable form of upset detection for the three upset classes, this is due to the generality added by using multiple classifiers. Combining the three classifiers does not add any significant complexity to the classification process. The multi-classifier system typically consisted of a support vector machine, an ensemble of decision trees (bagging or AdaBoost) and/or a logistic regression classifier. The given classification problem of upset detection has a maximum of 14 features that the classifiers are trained on, this is a relative small number of features. The support vector machine and logistic regression classifier therefore required little computational resources in classifying new observations. By limiting the maximum depth and number of the decision trees in the ensemble resulted in an ensemble classifier that required little computational resources. It is therefore viable to detect upset conditions with classification techniques on a flight computer within the timing requirements.

8.2 Recommendations for Future Work

This study was an investigation into the use of classification algorithms as an option for upset detection, as well as into the possibility of using inertial sensor data to add redundancy to anemometric sensor data. Estimation of anemometric data from inertial data is an alternative which can be used to add redundancy to the anemometric data. Suggested recommendations for future work is to do a quantitative comparison between classifier-based upset detection and to use estimated anemometric data within conventional upset detection systems.

Classifier performance on an on-board flight computer should also be investigated to ensure that the chosen classification algorithms are able to run within the timing requirements of the overall system. The ability of the classifier-based upset detection system to run within the timing requirements of an aircraft is an important aspect in ensuring reliable upset detection.

Multiple assumptions were made about the noise added to the sensor measurement used for classifier training and testing. A comprehensive study on sensor noise modelling should be conducted in order to improve the representativeness of the results obtained in this study.

It would also be useful to identify the top-performing classifier according to the specifications set, and to do a comprehensive classifier optimisation with the training configuration and training data used here.

Bibliography

[1] H. Ranter, "Airliner accident statistics 2006," *Aviation Safety Network*, 2007.

[2] J. E. Wilborn and J. V. Foster, "Defining commercial transport loss-of-control: A quantitative approach," in *AIAA atmospheric flight mechanics conference and exhibit*, 2004, p. 4811.

[3] C. M. Belcastro and J. V. Foster, "Aircraft loss-of-control accident analysis," in *Proceedings of AIAA Guidance, Navigation and Control Conference*, no. AIAA-2010-8004, Toronto, Canada, 2010.

[4] *Airworthiness standards: Transportation Category Airplanes*, Code of Federal Regulations Chapter 14, part 25, section 207. Std., 1964.

[5] P. Bala, E. Kupcis, and W. Fisher, "Aircraft stall warning system," US Patent 4,908,619.

[6] A. Heinsohn and D. Neary, "Aircraft stall warning indicator system based on rate of change of angle of attack," US Patent 3,839,699, October 1, 1974.

[7] S. Advani and J. Field, "Upset prevention and recovery training in flight simulators." in *AIAA Modelling and Simulation Technologies Conference*. National Aerospace Laboratory NLR, 2011.

[8] E. Groen, W. Ledegang, J. Field, H. Smaili, M. Roza, L. Fucke, S. Nooij, M. Goman, M. Mayrhofer, and L. Zaichik, "Supra-enhanced upset recovery simulation," in *AIAA Modeling and Simulation Technologies Conference*, 2012, pp. 2–5.

[9] W. Boyes, *Instrumentation reference book*. Butterworth-Heinemann, 2009.

[10] J. Perry, A. Mohamed, B. Johnson, and R. Lind, "Estimating angle of attack and sideslip under high dynamics on small UAVs," in *ION GNSS Conference*, 2008, pp. 16–19.

[11] M. Oosterom and R. Babuška, "Virtual sensor for the angle-of-attack signal in small commercial aircraft," in *IEEE International Conference on Fuzzy Systems*. IEEE, 2006, pp. 1396–1403.

[12] E. A. Morelli, "Real-time aerodynamic parameter estimation without air flow angle measurements," *Journal of Aircraft*, vol. 49, no. 4, pp. 1064–1074, 2012.

[13] V. Klein and E. A. Morelli, *Aircraft system identification: theory and practice*. American Institute of Aeronautics and Astronautics Reston, VA, USA, 2006.

[14] C. Ramprasadh and H. Arya, "Estimation of aerodynamic angles in a mini aerial vehicle under turbulent atmosphere," in *Proceedings of the Atmospheric Flight Mechanics Conference, AIAA, Portland, OR*, 2011.

[15] R. P. Collinson, *Introduction to avionics systems*. Springer Science & Business Media, 2013.

[16] E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, and J. Woollen, "The NCEP/NCAR 40-year reanalysis project," *Bulletin of the American meteorological Society*, vol. 77, no. 3, pp. 437–471, 1996.

[17] F. M. Hoblit, *Gust Loads on Aircraft: Concepts and Applications*, ser. AIAA Education. AIAA, 2001.

[18] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algotithms, and Applications*. Prentice Hall, 1996.

[19] D. T. Larose and C. D. Larose, *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[20] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[21] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[24] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[25] P. Z. Peebles, J. Read, and P. Read, *Probability, random variables, and random signal principles*. McGraw-Hill Boston, Mass, USA, 2001, vol. 3.

[26] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[27] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[28] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[29] J. Stewart, *Calculus*, 6th ed., C. V. Wagner, Ed. Cengage Learning, 2009.

[30] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 131–159, 2002.

[31] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[32] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[33] L. Rokach and O. Maimon, *Data mining with decision trees: theory and applications*. World scientific, 2014.

[34] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *ICML*, vol. 96, 1996, pp. 148–156.

[35] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal of the Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[36] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class AdaBoost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[37] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[38] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[39] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[40] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

Appendices

Appendix A

Input and Output Specifications

This appendix presents the sensors used for upset detection with the range and noise associated with each sensor. The range of the sensor measurements were taken from the ranges within the simulation model supplied by Airbus, this do not represent the absolute maximum and minimum measurements the sensor can measure, but rather the boundaries of the simulation model. The sensor noise were postulated form the sampling resolution and sampling frequency of the sensors.

A.1 Simulation Inputs and Initialisations

A.1.1 State Initialisation

The states initialised of the simulation model were the speed vector, angular rates and the aircraft attitude. The initial Euler attitude was converted to the quaternion attitude for use within the simulation. Tables A.1 and A.2 show the distributions of the initial states used to generate the random samples for the high angle of attack upset.

Table A.1: The Aircraft state initialisation for the widely distributed samples.

Name	**Wide Sample Distribution's Range**
Calibrated Airspeed (V_{cas})	$\mathcal{U}(180, 340)\ kn$
Angle of Attack (α)	$\mathcal{U}(0, 20)^\circ$
Side-Slip (β)	$\mathcal{N}(\mu = 0, \sigma^2 = 25)^\circ$
Roll Angle (ϕ)	$\mathcal{U}(-20, 20)^\circ$
Pitch Angle (θ)	$\mathcal{N}(\mu = \alpha, \sigma^2 = 100)^\circ$
Yaw Angle (ψ)	0°
Roll Rate (P)	$\mathcal{U}(-5, 5)^\circ/s$
Pitch Rate (Q)	$\mathcal{U}(-25, 25)^\circ/s$
Yaw Rate (R)	$\mathcal{U}(-5, 5)^\circ/s$

Table A.1 shows the wide sample distributions for the initial states and Table A.2 shows the tight normal sample distributions about the upset boundaries. The tight sample

distribution for the high angle of attack upset class had normal distribution with a variance of 0.25° around the critical angle of attack surface ($\alpha_{Crit}(Alt, V_{cas})$). The tightly normal distributed samples around the underspeed boundary ($Underspeed(Alt)$) had a variance of 12.25 kn. The samples distributed around the dynamic pitch upset boundary were a two dimensional Gaussian distribution with a variance of 0.25° for the angle of attack and 1°/s for the pitch rate. The dynamic pitch boundary was determined as were the critical angle of attack and the turning angle of attack are equal, $\alpha_{Crit}(Alt, V_{cas}) = \alpha_{turn}(Alt, V_{cas}, \alpha, Q)$. The turning angle of attack is the maximum angle of attack reached given a pitch rate and an angle of attack and maximum opposing elevator applied.

Table A.2: The Aircraft state initialisation of the tight normal distributed samples.

State Name	**Tight Sample Distribution about Upset Boundary Range**
High Angle of Attack Upset	
Angle of Attack (α)	$\mathcal{N}(\mu = \alpha_{Crit}(Alt, V_{cas})\ ,\ \sigma^2 = 0.25)^\circ$
Underspeed Upset	
Calibrated Airspeed (V_{cas})	$\mathcal{N}(\mu = Underspeed(Alt), \sigma^2 = 12.25)kn$
Dynamic Pitch Upset Upset	
Angle of Attack (α)	$\mathcal{N}(\mu = f(\alpha) = \sum_i \sum_j \sum_k (\alpha_{crit}(i,j) - \alpha_{turn}(i,j,\alpha,k) = 0),$ $\sigma^2 = 0.25)^\circ$
Pitch Rate (Q)	$\mathcal{N}(\mu = g(Q) = \sum_i \sum_j \sum_l (\alpha_{crit}(i,j) - \alpha_{turn}(i,j,l,Q) = 0),$ $\sigma^2 = 1)^\circ/s$

A.1.2 Simulation Inputs

The simulation inputs are shown in Table A.3, these inputs were kept the same for all the samples generated. The wind disturbance were taken as a uniform distribution between the 10th and 90th percentile of the global wind taken for 2015. The aerodynamic model does not take thrust as an input, the thrust were therefore taken as a constant value of 1.2°.

Table A.3: List of simulation inputs distribution specifications.

Name	**Wide Sample Distribution's Range**
Altitude (Alt)	$\mathcal{U}(0, 40\ 000)ft$
Aileron Deflection (δ_{ail})	$\mathcal{U}(-16, 16)^\circ$
Elevator Deflection (δ_{elv})	$\mathcal{U}(-16, 16)^\circ$
Rudder Deflection (δ_{rud})	$\mathcal{U}(-5, 5)^\circ$
Thrust	1.2°
Airbrakes	Disengaged
Longitudinal Wind (U_{wind})	$\mathcal{U}(10\%, 90\%)$
Lateral Wind (V_{wind})	$\mathcal{U}(10\%, 90\%)$
Normal Wind (W_{wind})	$\mathcal{U}(10\%, 90\%)$

A.2 Output Sensor Description

This section gives a description of the sensor measurements used in the upset detections system. The sensor resolution and sample rate are given with the variance of the assumed noise added to the calculated sensor measurements.

A.2.1 Anemometric Sensors

The three anemometric sensors used for upset detection were Calibrated Airspeed (Vcas), Angle of Attack (α) and Side-slip Angle (β). Table A.4 shows the anemometric sensor specifications and the variance of the noise added to the simulated sensor measurement. This noise assumes a faultless sensor operating under normal flight conditions. A faulty sensor might become biased and the sensor noise might increase.

Table A.4: The anemometric sensor description and specifications.

Sensor Name	**Analogue to Digital LSB Resolution (Δ)**	**Sampling Frequency (fs)**	**Noise Variance**
Calibrated Airspeed (V_{cas})	0.0625 kn	8.15	0.0106 kn
Angle of Attack (α)	0.0439°	16.31	0.0105°
Side-Slip (β)	0.0439°	16.31	0.0105°

A.2.2 Inertial Sensors

The inertial sensor specifications are shown in Table A.5. The assumption was made that the inertial sensors were not biased and that the random walk were negligibly small

as taken from the Honeywell product brochures for the QA2000 accelerometer and the GG1320AN Digital Laser Gyro.

Table A.5: The inertial sensor description and specifications.

Sensor Name	**Analogue to Digital LSB Resolution (Δ)**	**Sampling Frequency (fs)**	**Noise Variance**
Altitude (Alt)	0.125 ft	25	0.1302 ft
Roll Angle (ϕ)	0.005493°	50	0.0005°
Pitch Angle (θ)	0.005493°	50	0.0005°
Yaw Angle (ψ)	0.005493°	50	0.0005°
Flight Path Angle (γ)	0.005493°	25	0.0003°
Roll Rate (P)	0.003906°/s	50	0.0003°/s
Pitch Rate (Q)	0.003906°/s	50	0.0003°/s
Yaw Rate (R)	0.003906°/s	50	0.0003°/s
Longitudinal Total Body Acceleration (a_x)	0.000122 G	50	0.0000002 G
Lateral Total Body Acceleration (a_y)	0.000122 G	50	0.0000002 G
Normal Total Body Acceleration (a_z)	0.000122 G	50	0.0000002 G

A.2.3 Ground speed and Wind

The wind speed components are calculated form a wind magnitude and heading supplied by the INS. It is assumed that there were a small estimation error made by the INS to supply the estimates of the wind speed. Table A.6 shows the ground speed and wind speed noise specifications as taken for the underspeed upset detection.

Table A.6: The ground speed and wind speed estimate descriptions .

Sensor Name	**Analogue to Digital LSB Resolution (Δ)**	**Sampling Frequency (fs)**	**Noise Variance**
Ground Speed (V_{sol})	0.125 kn	25	0.1302 kn
Longitudinal Wind Speed ($U_{wind_{est}}$)	0.007813 kn	12.5	0.0003 kn
Lateral Wind Speed ($V_{wind_{est}}$)	0.007813 kn	12.5	0.0003 kn

Appendix B

Classifier Training Parameter Search

The classifier training configuration parameter search for each of the classifiers are described. A random search was conducted within the classifier training parameter space to determine the sensitivity of the different classifiers to a change in training parameters. This is used to determine the amount of tuning necessary to achieve a high accuracy for a given classifier. The parameter ranges and sampling strategies are given in this appendix. The default training parameters were included within the parameter search space.

B.1 Support Vector Machine

B.1.1 Default Training Parameters

The default Scikit-Learn training parameters are shown in Table B.1.

Table B.1: SVM's default training parameter description.

Parameter Name	**Parameter Description**	**Parameter Value**
C	Penalty Parameter	1.0
gamma	Kernel Coefficient	1/features
kernel	Kernel type used	Radial Bases Function
tol	Tolerance for stopping criterion	1e-3

B.1.2 Parameter Search

Table B.2 shows the SVM's training parameter and the parameter ranges searched.

Table B.2: SVM's training parameter random search description.

Parameter Name	Parameter Search Range	Search description
C	[0.5 10.5]	Uniformly sampled
gamma	[0 0.5]	Uniformly sampled
kernel	[linear, poly, rbf, sigmoid]	Randomly drawn
tol	[0.0001 0.01]	Uniformly sampled

B.2 Decision Tree

B.2.1 Default Training Parameters

The default Scikit-Learn training parameters for the Decision tree classifier are shown in Table B.3.

Table B.3: DT's default training parameter description.

Parameter Name	Parameter Description	Parameter Value
Criterion	The function to measure the quality of a split	gini
Max Features	The maximum features to search for a split	All the Features
Max Depth	The maximum amount of layers in the tree	No limit
Min Samples Split	The minimum number of samples required to split an internal node	2
Min Samples Node	The minimum number of samples required to be at a leaf node	1

B.2.2 Parameter Search

The training parameters searched are shown in Table B.4. The parameter ranges are shown with the sampling strategy used.

Table B.4: DT's training parameter random search description.

Parameter Name	**Parameter Search Range**	**Search description**
Criterion	[Gini,Entropy]	Randomly Drawn
Max Features	[1, 11]	Uniform integer distribution
Max Depth	[3,None]	Randomly Drawn
Min Samples Split	[1, 11]	Uniform integer distribution
Min Samples Node	[1, 11]	Uniform integer distribution

B.3 Naïve Bayes

The naïve bayes classifier does not have any training parameters, it assumes feature independence and a Gaussian distribution spread of the two classes. The were therefore no parameters searched for the naïve bayes classifier.

B.4 Nearest Neighbour

B.4.1 Default Training Parameters

The default Scikit-Learn training parameters for the nearest neighbour classifier are shown in Table B.5.

Table B.5: knn's default training parameter description.

Parameter Name	**Parameter Description**	**Parameter Value**
n neighbours	Number of neighbours to use	5
Leaf size	Leaf size passed to Ball tree or KD tree	30
Weights	Weight function used in prediction	Uniform

B.4.2 Parameter Search

The nearest neighbour parameter ranges searched are shown in Table B.6.

Table B.6: knn's training parameter random search description.

Parameter Name	Parameter Search Range	Search description
n neighbours	[5 50]	Uniform integer distribution
Leaf size	[20 100]	Uniform integer distribution
Weights	[Uniform, Distance]	Randomly Drawn

B.5 AdaBoost

B.5.1 Default Training Parameters

The default AdaBoost training parameters used by Scikit-Learn are shown in Table B.7.

Table B.7: AdaBoost's default training parameter description.

Parameter Name	Parameter Description	Parameter Value
Base Estimator	The weak learner from which the ensemble is built	Decision Tree
n Estimators	The number of weak learners used	50
Learning Rate	Learning rate shrinks the contribution of each classifier	1
Algorithm	The boosting algorithm used	SAMME.R

B.5.2 Parameter Search

The Scikit-Learn AdaBoost classifier's training parameter ranges used in the random search are shown in Table B.8.

Table B.8: AdaBoost's training parameter random search description.

Parameter Name	Parameter Search Range	Search description
Base Estimator	[Decision Tree]	Randomly Drawn
n Estimators	[20 200]	Uniform integer distribution
Learning Rate	[0 1]	Uniformly Sampled
Algorithm	[SAMME SAMME.R]	Randomly Drawn

B.6 Random Forest

B.6.1 Default Training Parameters

The default random forest Scikit-Learn parameters are shown in Table B.9.

Table B.9: RF's default training parameter description.

Parameter Name	Parameter Description	Parameter Value
n Estimators	The number of decision trees in ensemble	10
Criterion	The splitting criterion used	gini
Max depth	The maximum depth of the trees grown	No limit
Min Samples Split	Minimum number of samples required to split a node	2
Min Samples Leaf	Minimum number of samples in leaf nodes	1

B.6.2 Parameter Search

The training parameters searched for the random forest classifier are shown in Table B.10.

Table B.10: RF's training parameter random search description.

Parameter Name	Parameter Search Range	Search description
n Estimators	[10 100]	Uniform integer distribution
Criterion	[gini,entropy]	Randomly Drawn
Max Depth	[3 10]	Uniform integer distribution
Min Samples Split	[1 11]	Uniform integer distribution
Min Samples Leaf	[1 11]	Uniform integer distribution

B.7 Bagging Classifier

B.7.1 Default Training Parameters

The default Bagging classifier training parameters used are shown in Table B.11.

Table B.11: Bagging's default training parameter description.

Parameter Name	**Parameter Description**	**Parameter Value**
Base Estimator	The weak learner used in ensemble	Decision tree
n Estimators	The number of weak learners used in ensemble	10
Max Samples	Maximum amount of samples drawn	100% of samples
Max Features	The number of features drawn for training	100% of features

B.7.2 Parameter Search

The Bagging classifier's training parameter searched are shown in Table B.12.

Table B.12: Bagging's training parameter random search description.

Parameter Name	**Parameter Search Range**	**Search description**
Base Estimator	[Decision Tree]	Randomly Drawn
n Estimators	[10 100]	Uniform Integer Distribution
Max Samples	[30% 70%]	Uniformly distributed
Max Features	[30% 70%]	Uniformly distributed

B.8 Multi Classifier System

The multi classifier system is a combination of the three top performing classifiers. The classifier fusion is taken as the majority vote of equally weighted classifiers. There is therefore no additional training done by the multi classifier system that can be searched.

B.9 Logistic Regression Classifier

B.9.1 Default Training Parameters

Table B.9 show the default Scikit-Learn parameters used for the logistic regression classifier.

Table B.13: LR's default training parameter description.

Parameter Name	**Parameter Description**	**Parameter Value**
C	Inverse of regularisation strength	1
Solver	Gradient decent optimisation algorithm	liblinear

B.9.2 Parameter Search

The logistic regression training configuration parameters searched are shown in Table B.10

Table B.14: LR's training parameter random search description.

Parameter Name	**Parameter Search Range**	**Search description**
C	[0.5 10]	Uniformly Distibution
Solver	[newton-cg, lbfgs, liblinear]	Randomly Drawn

Appendix C

Additional High Angle of Attack Upset Classification Results

C.1 Classifier Learning Curves

The learning curves show the classifier accuracy trained on an increasing number of training samples. This is used to determine that the training dataset were large enough to ensure a sufficiently trained classifier for high angle of attack upset detection. All classifiers were trained with sufficiently large datasets to ensure fully trained classifiers. This is clear from the small confidence intervals at large datasets, and the little increase in classifier accuracy at higher training sample numbers. Tenfold cross-validation was done on the different sizes of training datasets. The 95% confidence intervals were calculated from the ten fold cross validation.

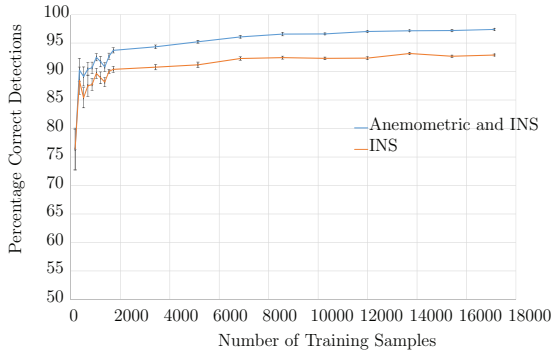

Figure C.1: The SVM's learning curve for both sensor cases.

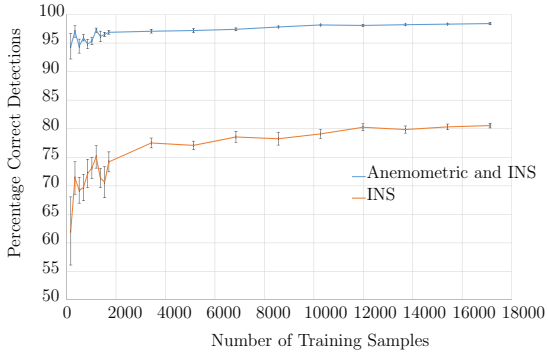

Figure C.2: The DT's learning curve for both sensor cases.

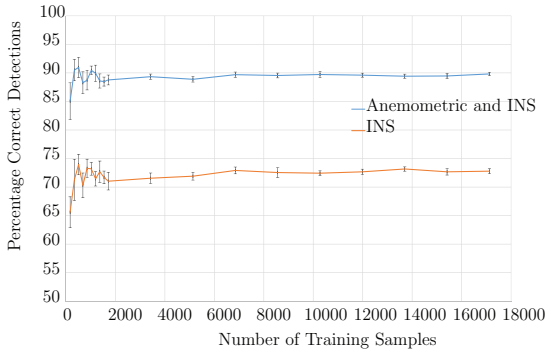

Figure C.3: The NB's learning curve for both sensor cases.

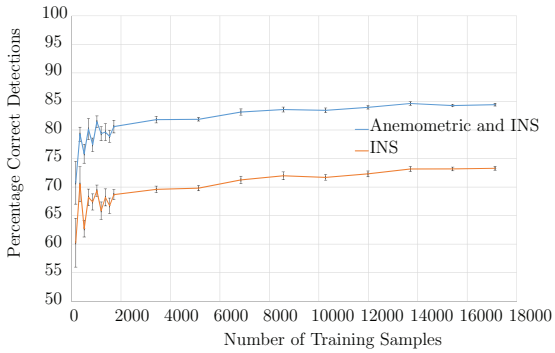

Figure C.4: The knn's learning curve for both sensor cases.

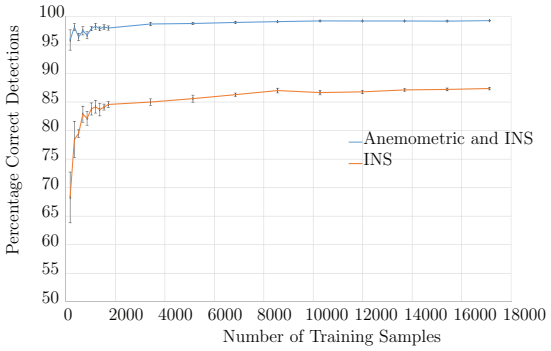

Figure C.5: The AdaBoost's learning curve for both sensor cases.

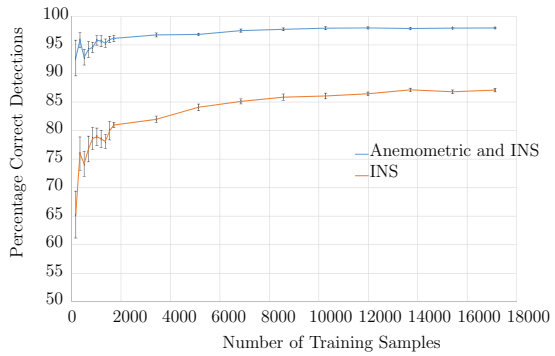

Figure C.6: The RF's learning curve for both sensor cases.

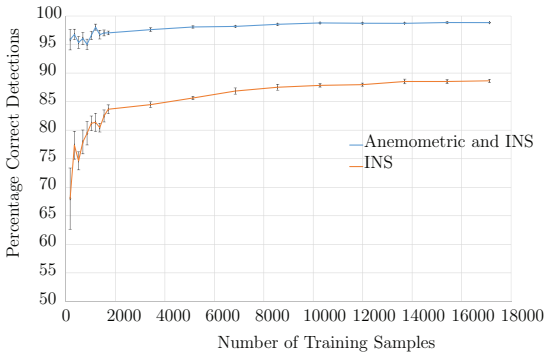

Figure C.7: The Bagging's learning curve for both sensor cases.

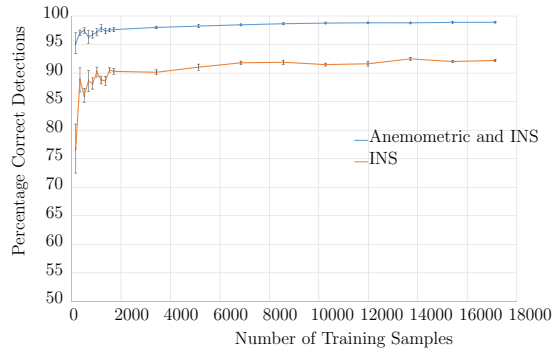

Figure C.8: The MCS's learning curve for both sensor cases.

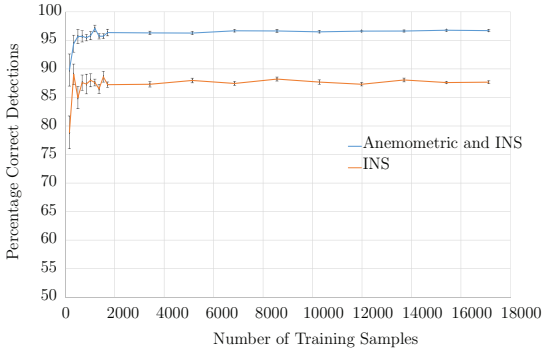

Figure C.9: The LR's learning curve for both sensor cases.

C.2 Receiver Operating Characteristic Curves for Classifiers

The receiver operating characteristic curves are shown for all the classifiers trained to detect high angle of attack upset. The ROC curves for both sensor cases are shown on the same figure. These curves offer insight into the false alarms and missed detections generated at different thresholds on the probability outputs of the classifiers. The classifiers are required to perform at an equal error rate, the ROC curves are used to determine the desired threshold to ensure an equal error rate. It is clear from the ROC curves shown that the EER's threshold was typically near a probability of 0.5. The ROC curves can be used to bias the classifiers towards missed detections to reduce the amount of false alarms generated.

The DT and AdaBoost classifiers are rule based classifiers that are trained to perform at an equal error rate at a threshold of 0.5, the ROC curves are therefore not smooth for these two classifiers.

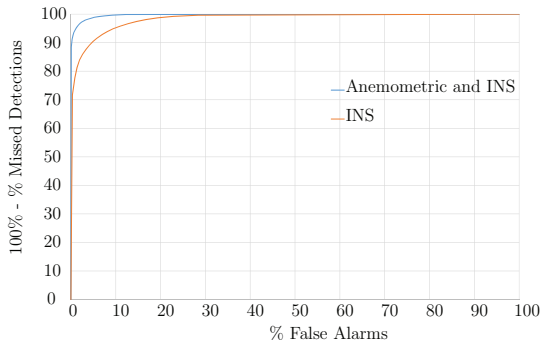

Figure C.10: The SVM's ROC curve for both sensor cases.

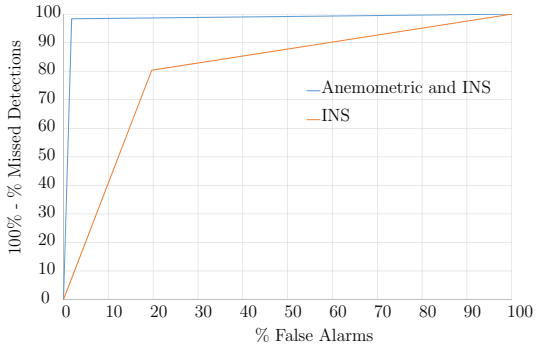

Figure C.11: The DT's ROC curve for both sensor cases.

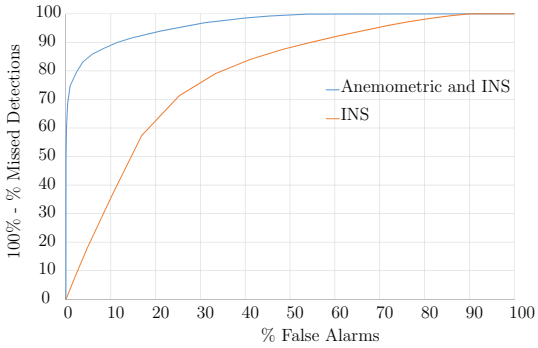

Figure C.12: The NB's ROC curve for both sensor cases.

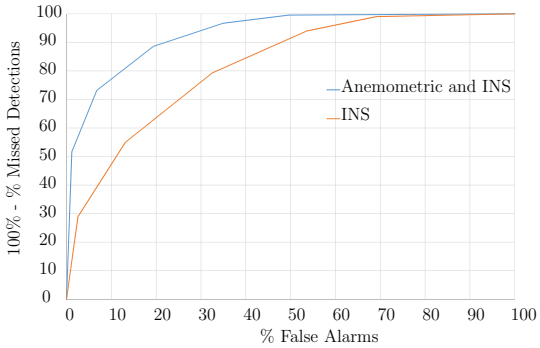

Figure C.13: The knn's ROC curve for both sensor cases.

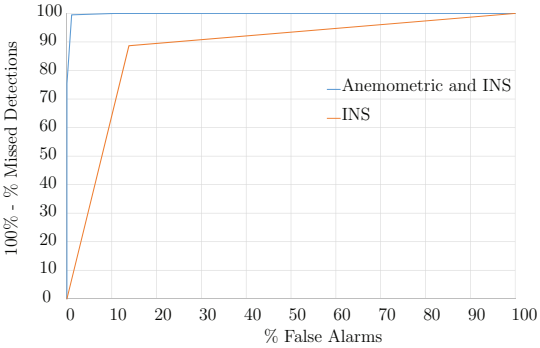

Figure C.14: The AdaBoost's ROC curve for both sensor cases.

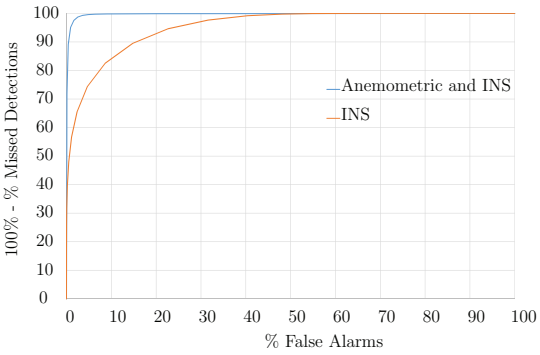

Figure C.15: The RF's ROC curve for both sensor cases.

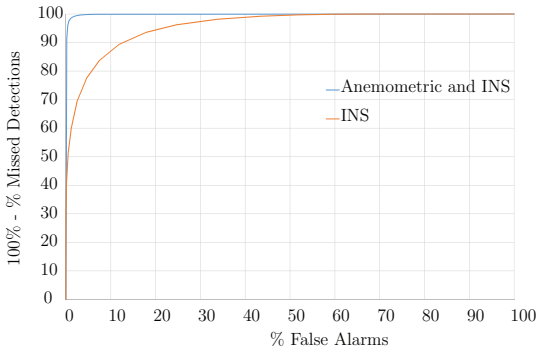

Figure C.16: The Bagging's ROC curve for both sensor cases.

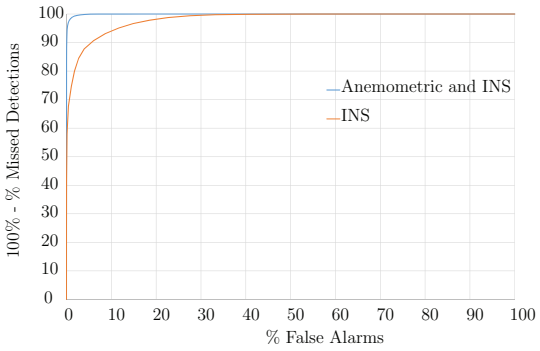

Figure C.17: The MCS's ROC curve for both sensor cases.

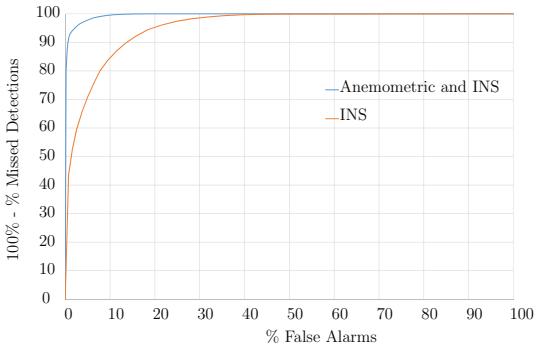

Figure C.18: The LR's ROC curve for both sensor cases.

C.3 Classifier Accuracy at Multiple Noise Levels

The classifier accuracies at different noise levels on the angle of attack sensor are shown in Figure C.19 to C.27. The noise on the angle of attack was assumed to be a zero mean gaussian distribution. The standard deviation of the noise were varied between 0 degrees to 3 degrees and the percentage of correct detections were calculated.

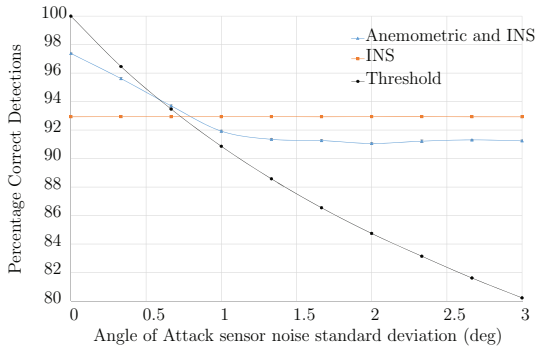

Figure C.19: The SVM's accuracy at different angle of attack sensor noise levels.

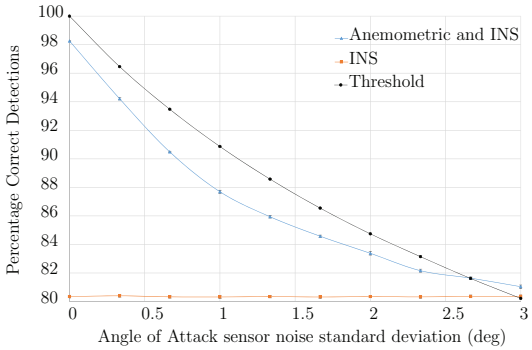

Figure C.20: The DT's accuracy at different angle of attack sensor noise levels.

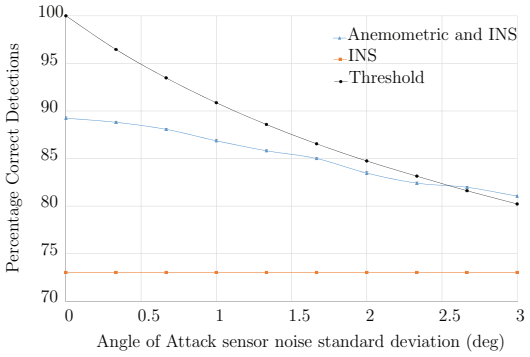

Figure C.21: The NB's accuracy at different angle of attack sensor noise levels.

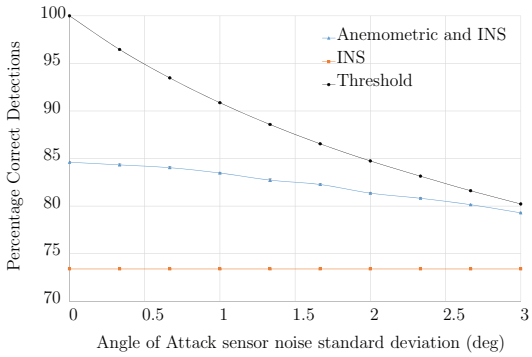

Figure C.22: The knn's accuracy at different angle of attack sensor noise levels.

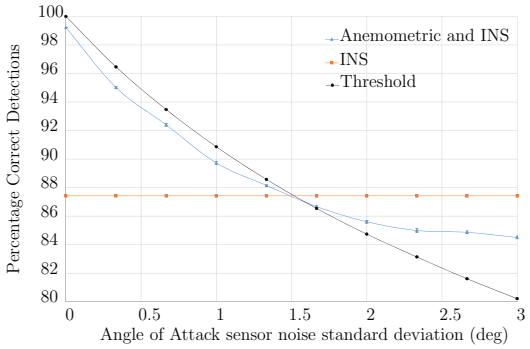

Figure C.23: The AdaBoost's accuracy at different angle of attack sensor noise levels.

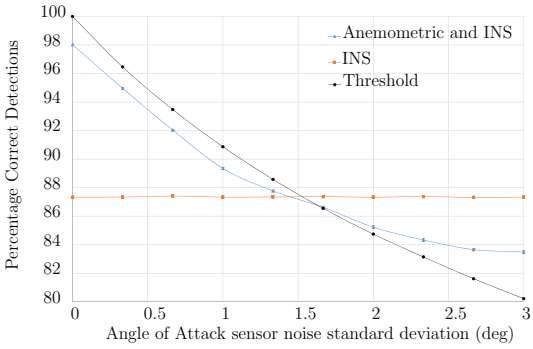

Figure C.24: The RF's accuracy at different angle of attack sensor noise levels.

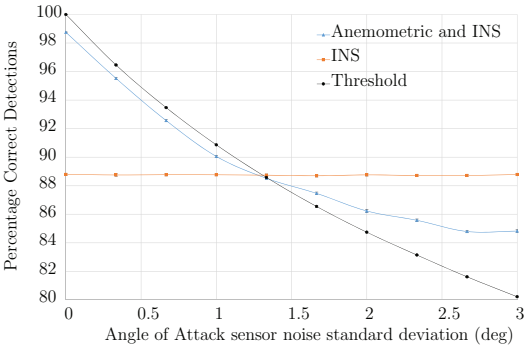

Figure C.25: The Bagging's accuracy at different angle of attack sensor noise levels.

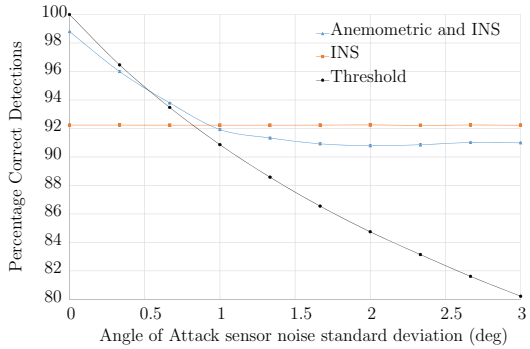

Figure C.26: The MCS's accuracy at different angle of attack sensor noise levels.

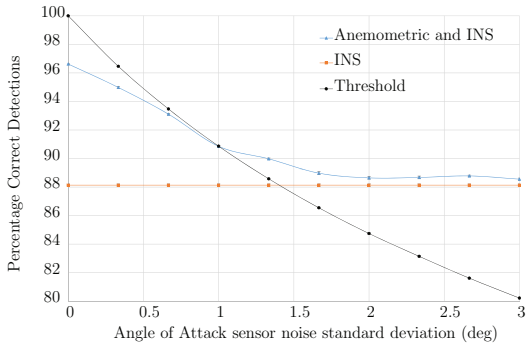

Figure C.27: The LR's accuracy at different angle of attack sensor noise levels.

C.4 False Alarm Distribution

This section shows the angle of attack distribution of the false alarms generated by classifiers as well as the angle of attack distribution of an Airbus A330 during normal flight at $17500ft$ and $240kn$ under moderate turbulence.

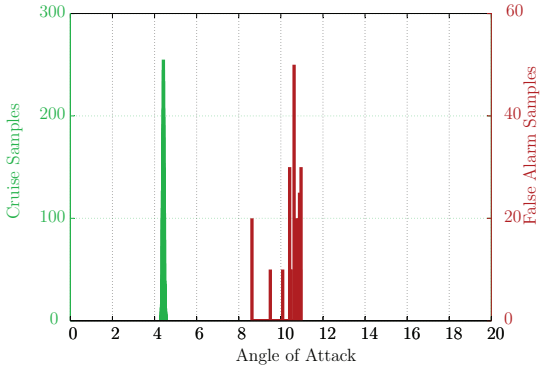

Figure C.28: SVM's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

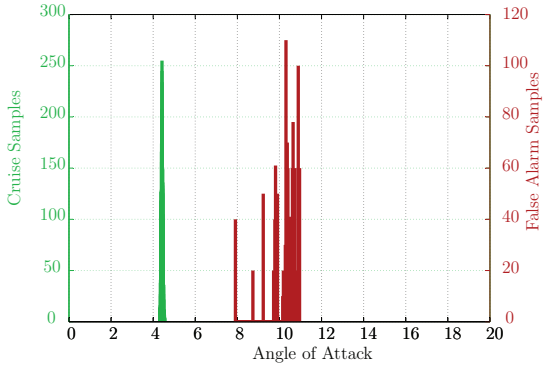

Figure C.29: SVM's false alarm and cruise angle of attack distribution for only the inertial sensors available.

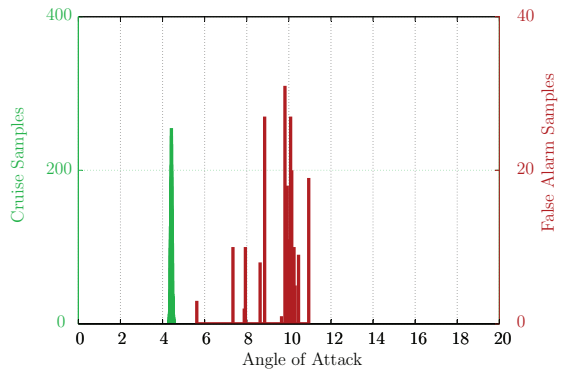

Figure C.30: DT's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

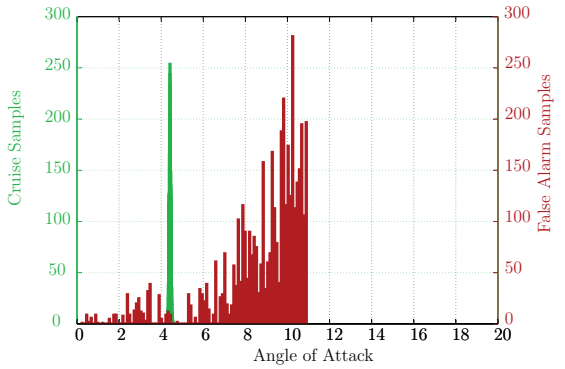

Figure C.31: DT's false alarm and cruise angle of attack distribution for only the inertial sensors available.

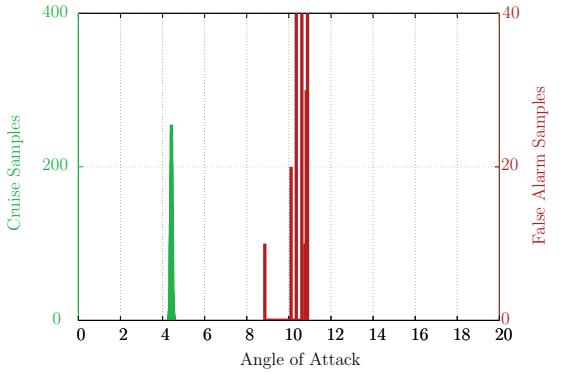

Figure C.32: NB's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

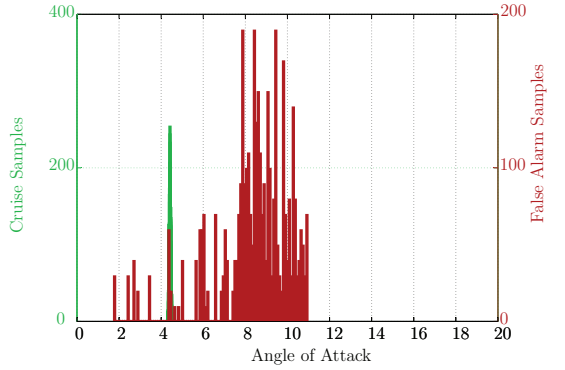

Figure C.33: NB's false alarm and cruise angle of attack distribution for only the inertial sensors available.

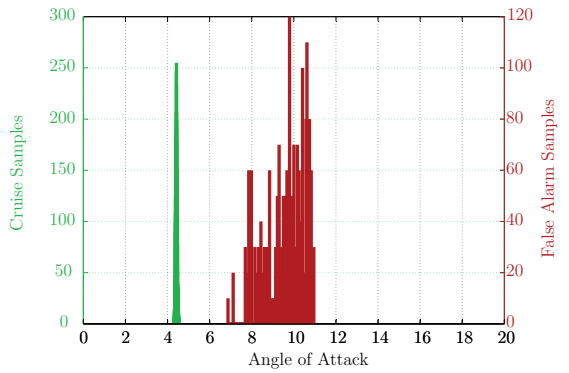

Figure C.34: knn's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

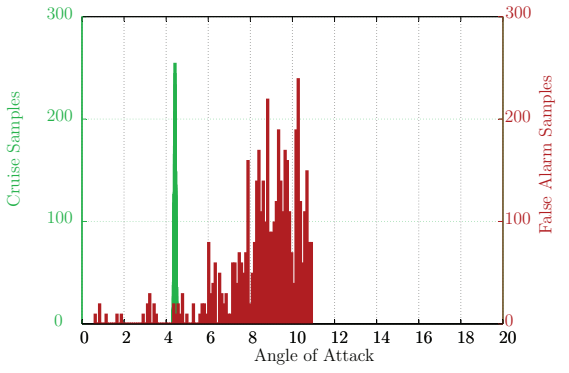

Figure C.35: knn's false alarm and cruise angle of attack distribution for only the inertial sensors available.

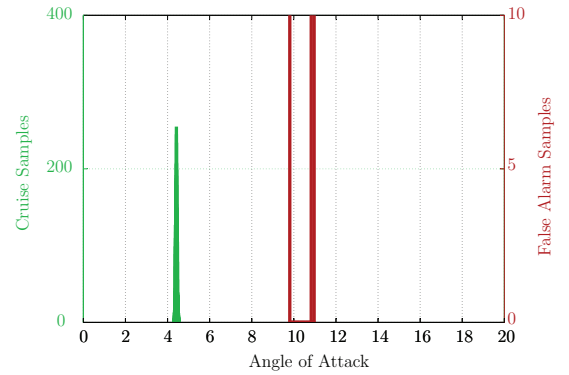

Figure C.36: AdaBoost's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

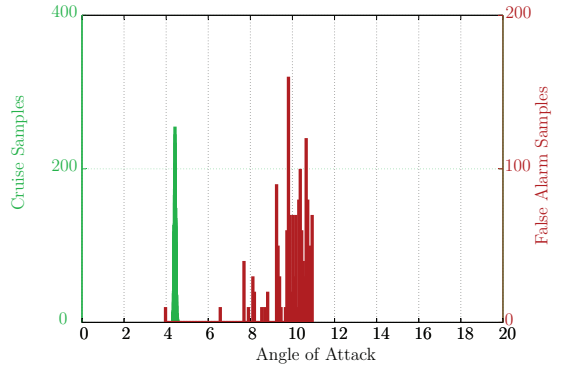

Figure C.37: AdaBoost's false alarm and cruise angle of attack distribution for only the inertial sensors available.

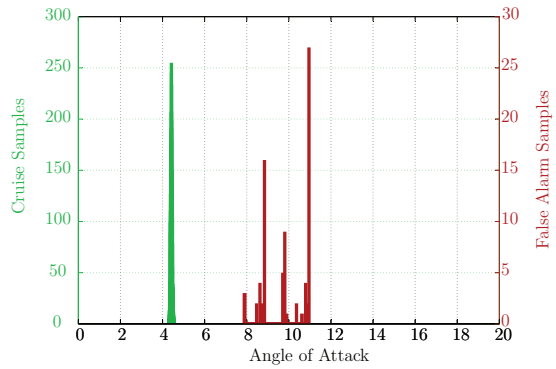

Figure C.38: RF's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

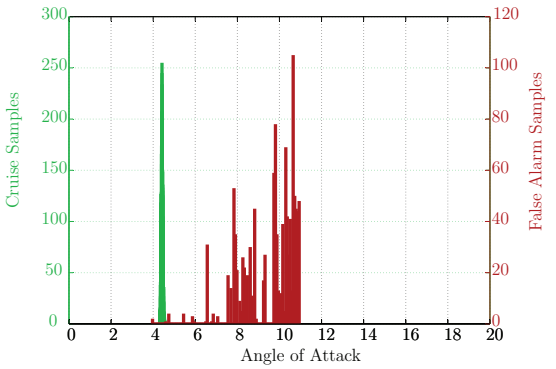

Figure C.39: RF's false alarm and cruise angle of attack distribution for only the inertial sensors available.

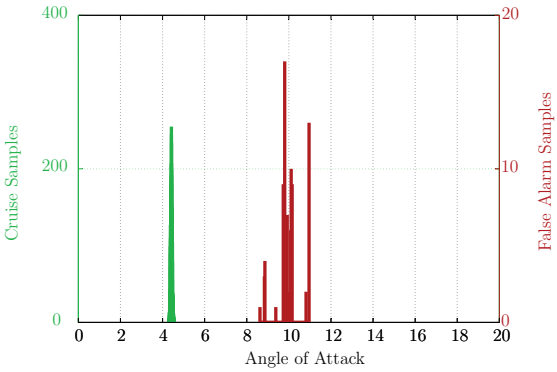

Figure C.40: Bagging's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

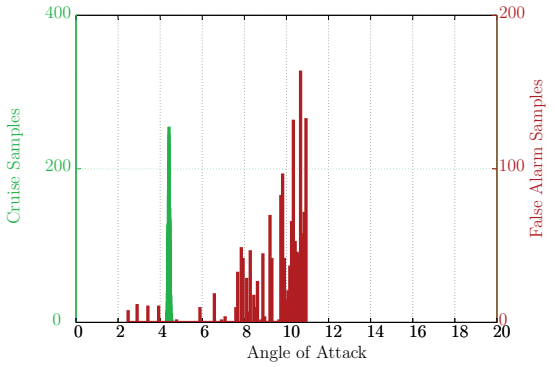

Figure C.41: Bagging's false alarm and cruise angle of attack distribution for only the inertial sensors available.

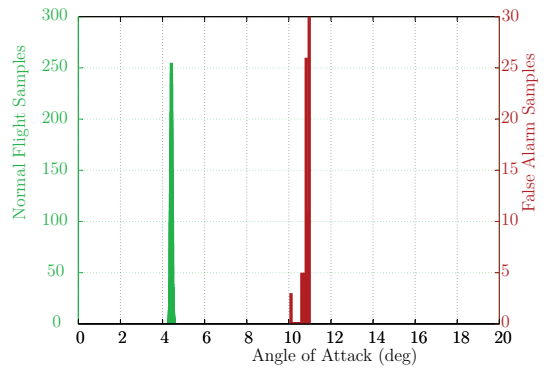

Figure C.42: MCS's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

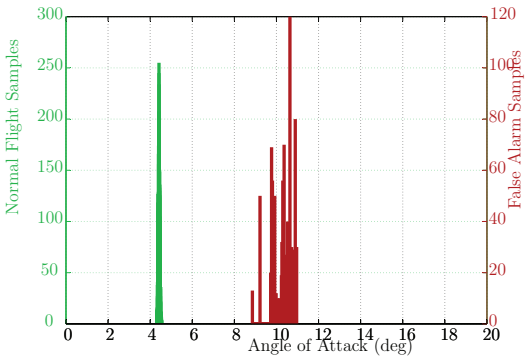

Figure C.43: MCS's false alarm and cruise angle of attack distribution for only the inertial sensors available.

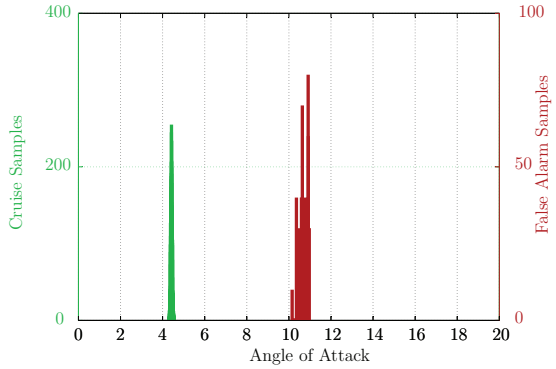

Figure C.44: LR's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

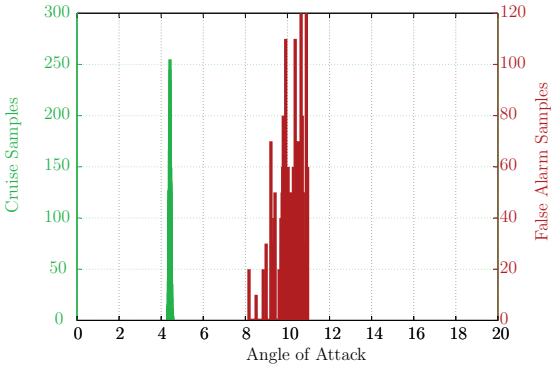

Figure C.45: LR's false alarm and cruise angle of attack distribution for only the inertial sensors available.

C.5 Classifier Accuracy during an Upset Manoeuvre

This section shows the false alarm and missed detections generated by the classifier during a high angle of attack validation manoeuvre. The manoeuvre performed tested three aspects of the classification based upset detection function. Classifiers were tested during manoeuvres that fell within the normal flight operational envelope. The classifiers were secondly tested om manoeuvres that resulted in the aircraft entering the upset domain. Manoeuvres recovering from an upset event were lastly tested. The position of the false alarms and missed detections from the upset boundary were evaluated. The manoeuvres were used to identify whether the sensor measurements in the training dataset were representative of typical manoeuvres within the upset and normal cruise domains.

The blue line in the figures shows the angle of attack trajectory during the manoeuvre. The black line show the upset boundary for the given altitude and airspeeds, a one degree region from the upset boundary in each direction are indicated with the two red lines.

It is preferable if the false alarms and missed detections fall within this region. The false alarms are indicated with red dots and the missed detections are indicated with green dots. The AdaBoost and LR classifiers were the only classifiers that produced false alarms and missed detections that fell within the $\pm 1^\circ$ region for both sensor cases. The majority of the classifiers' false alarms and missed detections when the anemometric and INS sensors were within the $\pm 1^\circ$ region from the upset boundary.

The majority of the rule based classifiers (DT,RF,Bagging), had poor accuracy for the case where only the INS data was available, with false alarms well within the normal cruise domain of the aircraft.

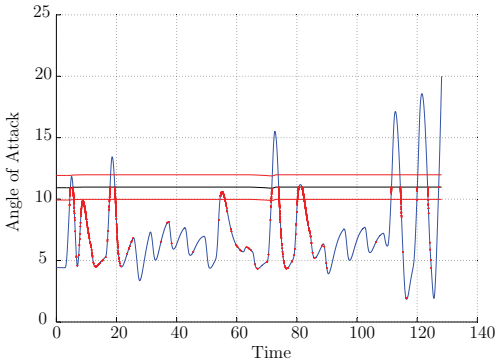

Figure C.46: SVM's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

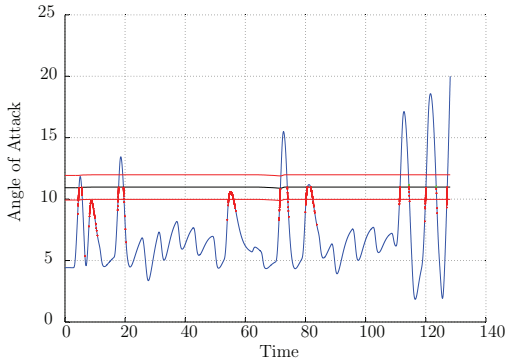

Figure C.47: SVM's false alarm and missed detections during a manoeuvre with only the inertial sensors available

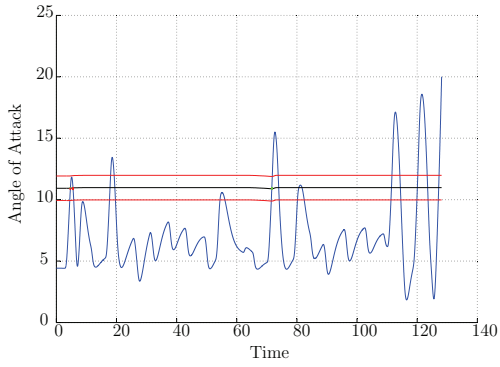

Figure C.48: DT's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

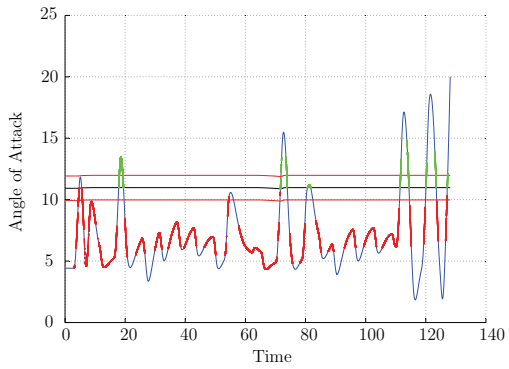

Figure C.49: DT's false alarm and missed detections during a manoeuvre with only the inertial sensors available

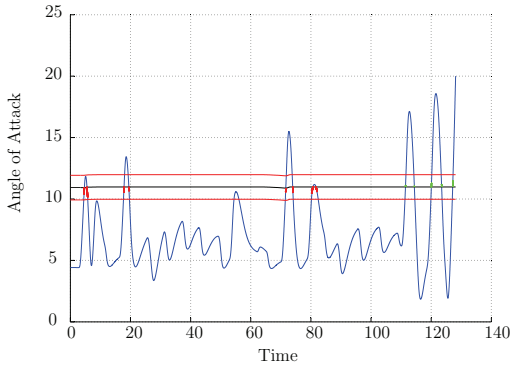

Figure C.50: NB's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

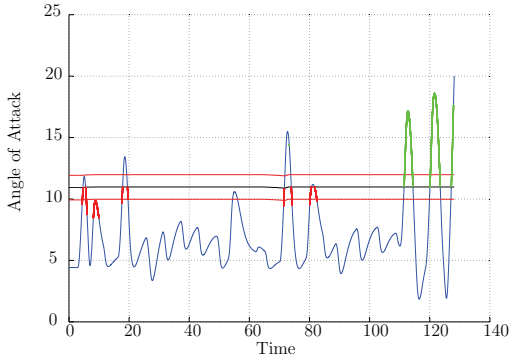

Figure C.51: NB's false alarm and missed detections during a manoeuvre with only the inertial sensors available

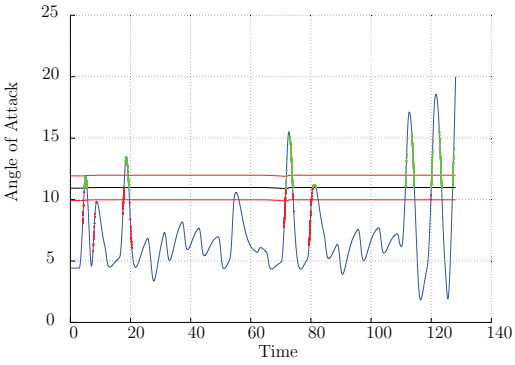

Figure C.52: knn's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

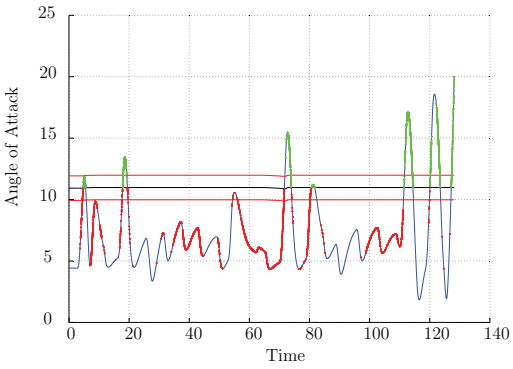

Figure C.53: knn's false alarm and missed detections during a manoeuvre with only the inertial sensors available

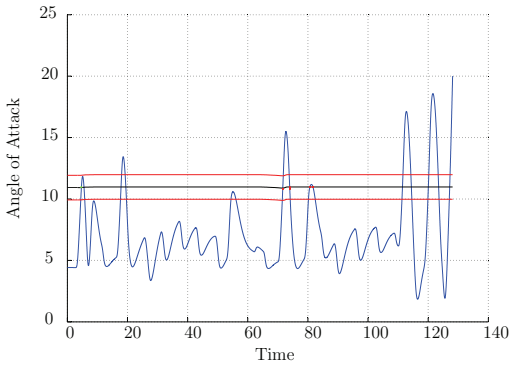

Figure C.54: AdaBoost's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

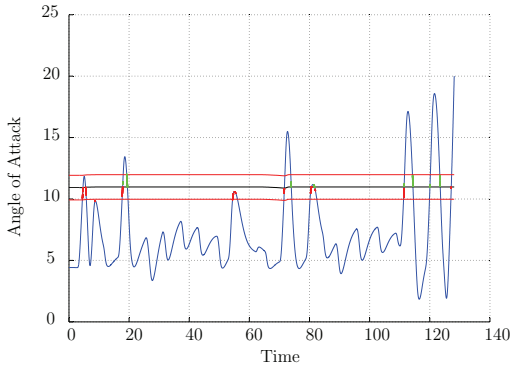

Figure C.55: AdaBoost's false alarm and missed detections during a manoeuvre with only the inertial sensors available

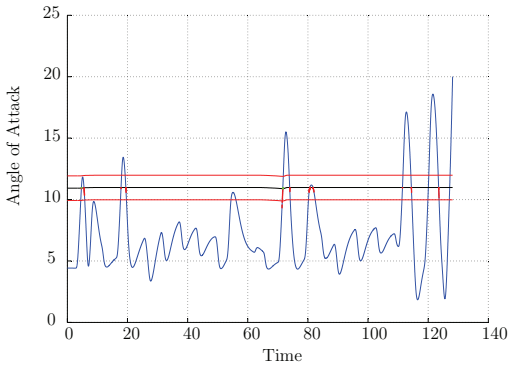

Figure C.56: RF's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

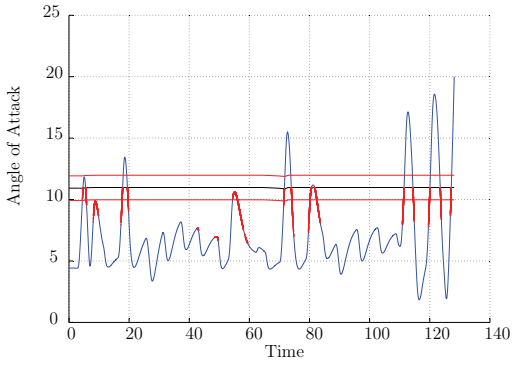

Figure C.57: RF's false alarm and missed detections during a manoeuvre with only the inertial sensors available

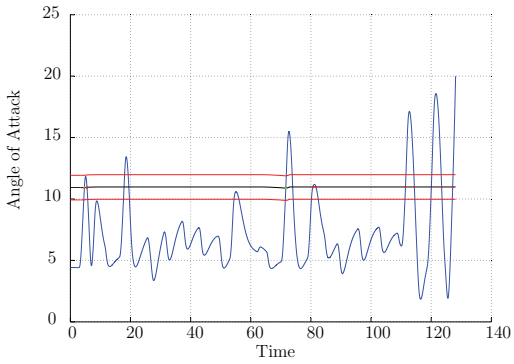

Figure C.58: Bagging's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

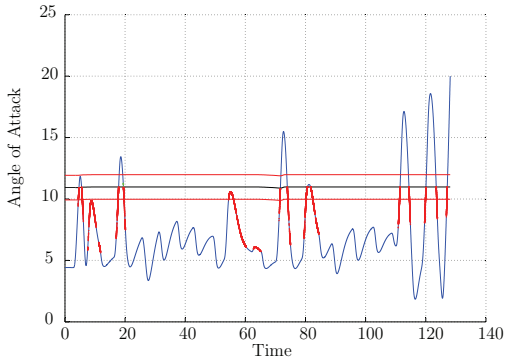

Figure C.59: Bagging's false alarm and missed detections during a manoeuvre with only the inertial sensors available

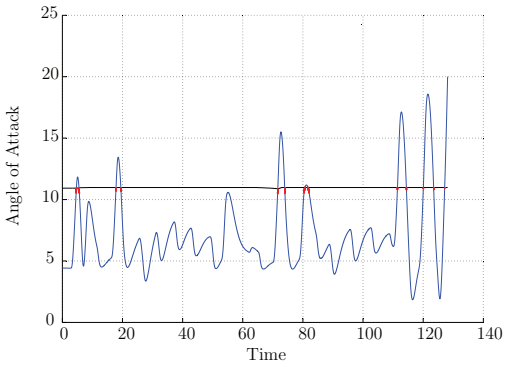

Figure C.60: MCS's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

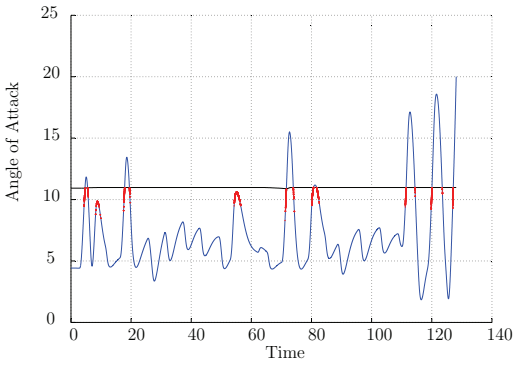

Figure C.61: MCS's false alarm and missed detections during a manoeuvre with only the inertial sensors available

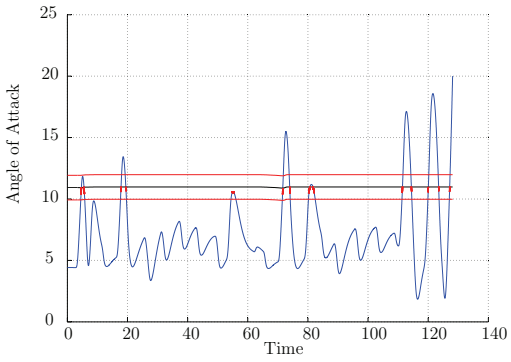

Figure C.62: LR's false alarm and missed detections during a manoeuvre with the anemometric and inertial sensors available.

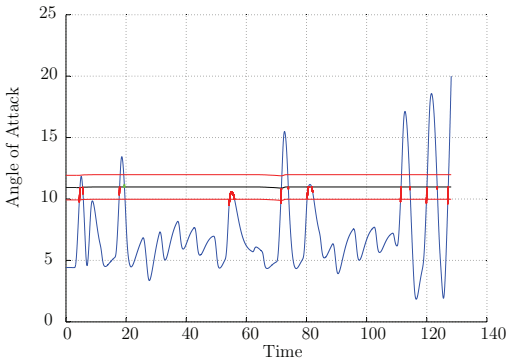

Figure C.63: LR's false alarm and missed detections during a manoeuvre with only the inertial sensors available

Appendix D

Additional Underspeed Upset Classification Results

D.1 Underspeed Upset Detection Classifier Learning Curves

This section shows the learning curves for the classifiers used for underspeed upset detection. The classifiers were trained with datasets increasing in size, and the accuracy were recorded. The learning curves are shown for the three sensor cases namely, anemometric and inertial sensors available, only inertial sensors available and inertial with estimates of the ground and wind speed available. These curves were used to ensure that the classifiers were sufficiently trained.

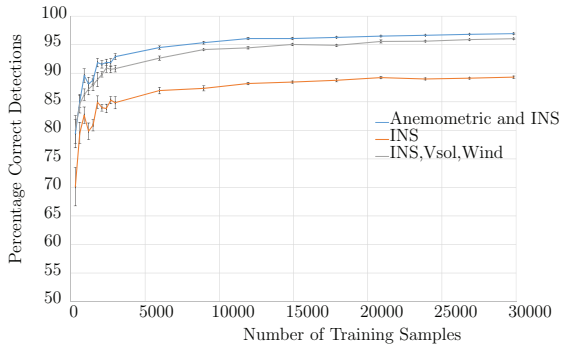

Figure D.1: The SVM's learning curve for both sensor cases.

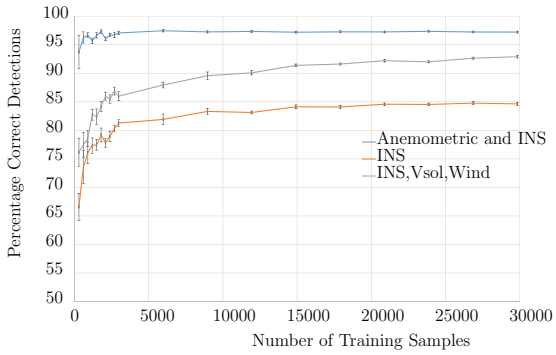

Figure D.2: The DT's learning curve for both sensor cases.

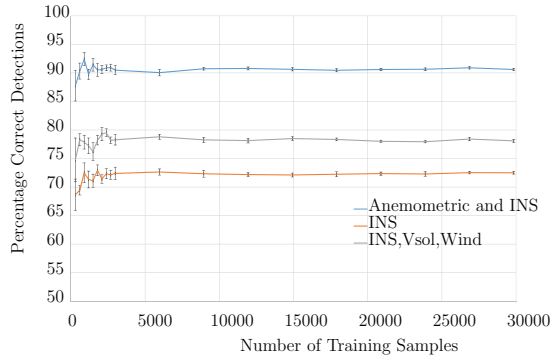

Figure D.3: The NB's learning curve for both sensor cases.

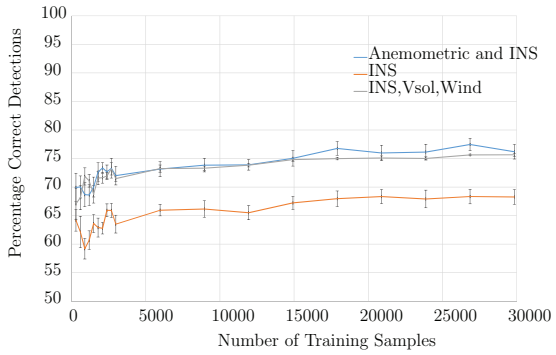

Figure D.4: The knn's learning curve for both sensor cases.

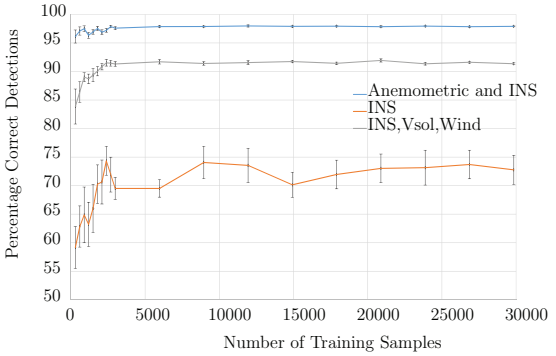

Figure D.5: The AdaBoost's learning curve for both sensor cases.

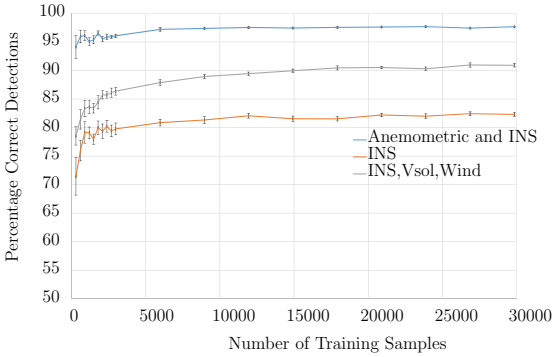

Figure D.6: The RF's learning curve for both sensor cases.

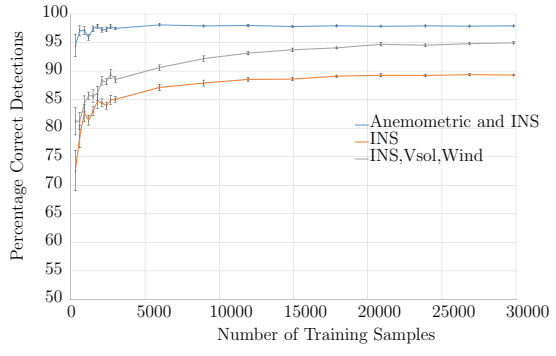

Figure D.7: The Bagging's learning curve for both sensor cases.

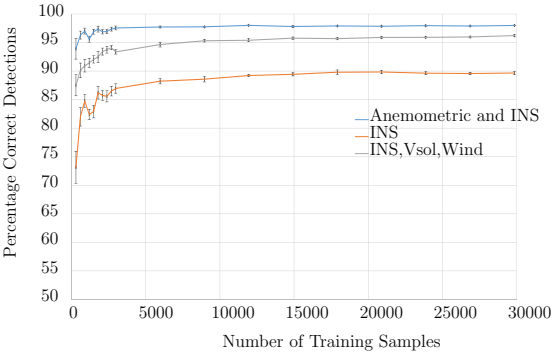

Figure D.8: The MCS's learning curve for both sensor cases.

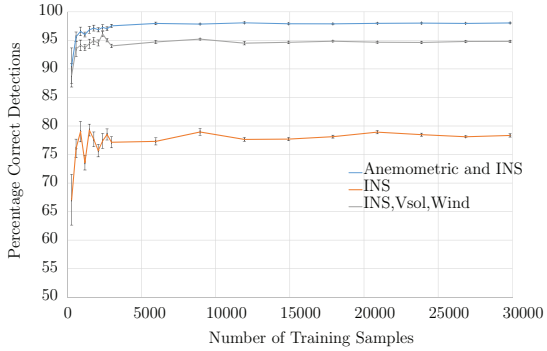

Figure D.9: The LR's learning curve for both sensor cases.

D.2 Receiver Operator Characterisation Curves for Classifiers

This section shows the ROC curves for the different classifiers used in underspeed upset detection. The ROC curves for all three sensor cases are shown in Figure D.10 to Figure D.18. The false alarms and missed detections were determined at increasing probability thresholds in order to determine the ROC curves.

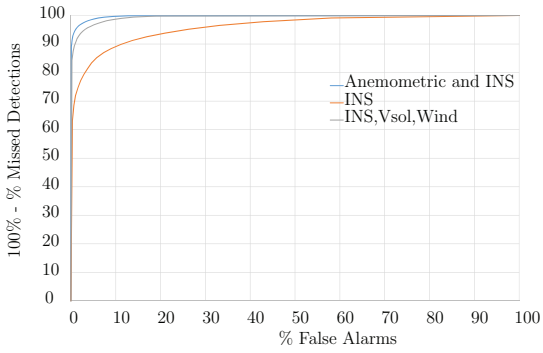

Figure D.10: The SVM's ROC curve for both sensor cases.

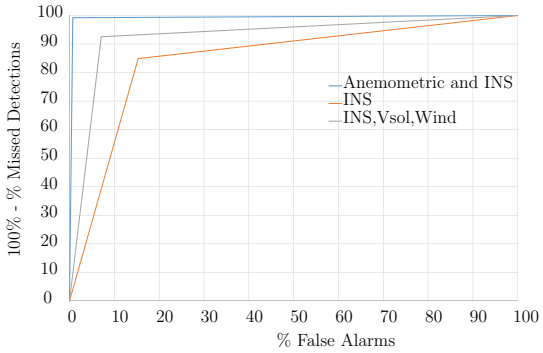

Figure D.11: The DT's ROC curve for both sensor cases.

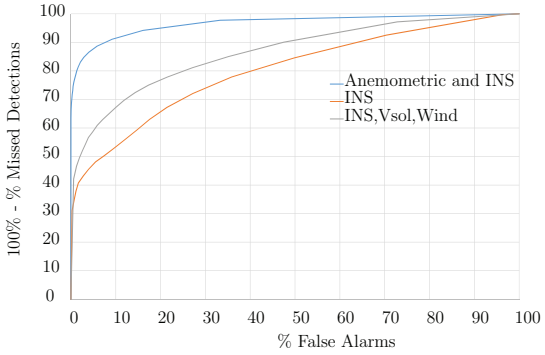

Figure D.12: The NB's ROC curve for both sensor cases.

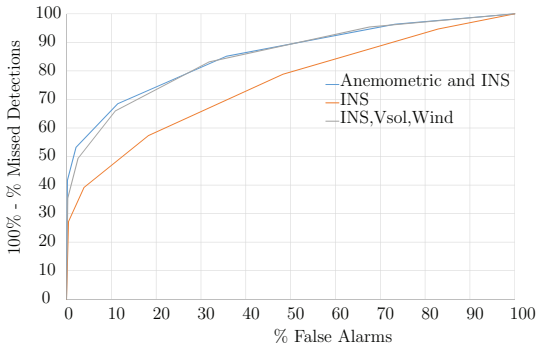

Figure D.13: The knn's ROC curve for both sensor cases.

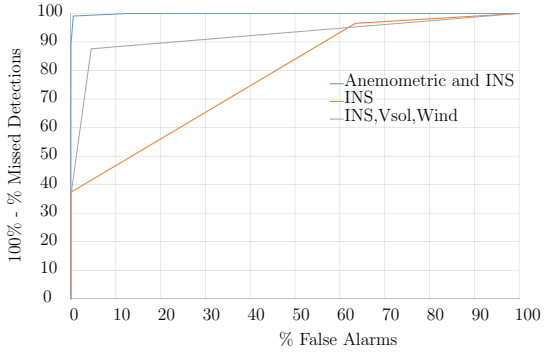

Figure D.14: The AdaBoost's ROC curve for both sensor cases.

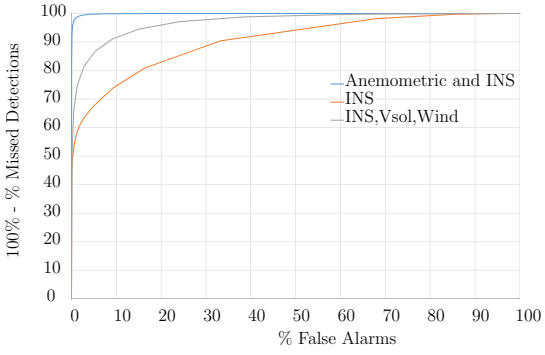

Figure D.15: The RF's ROC curve for both sensor cases.

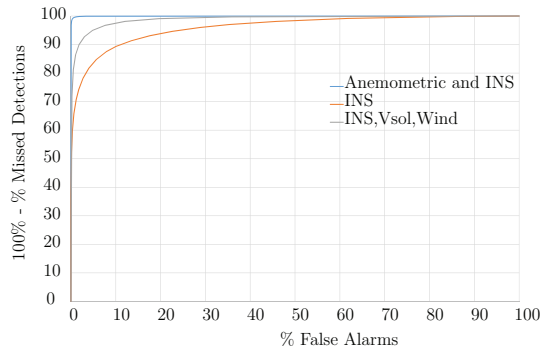

Figure D.16: The Bagging's ROC curve for both sensor cases.

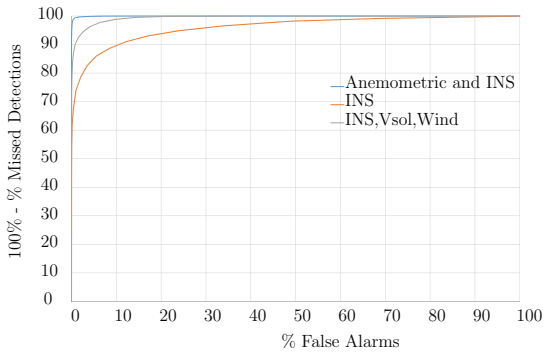

Figure D.17: The MCS's ROC curve for both sensor cases.

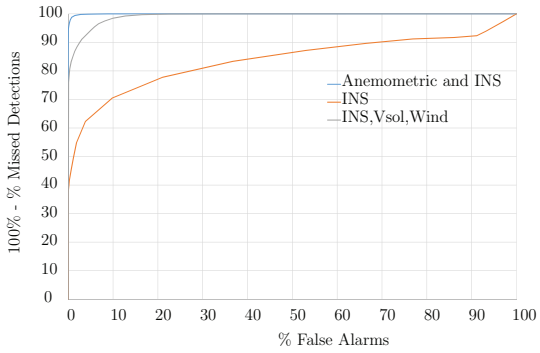

Figure D.18: The LR's ROC curve for both sensor cases.

D.3 Underspeed Upset Detections at Different Airspeed Noise Levels

The classifier accuracies at different noise levels on the airspeed sensor are shown in Figure D.19 to D.27. The noise on the airspeed sensor was assumed to be a zero mean gaussian distribution. The standard deviation of the noise were varied between $0kn$ to $30kn$ and the percentage correct detections were calculated.

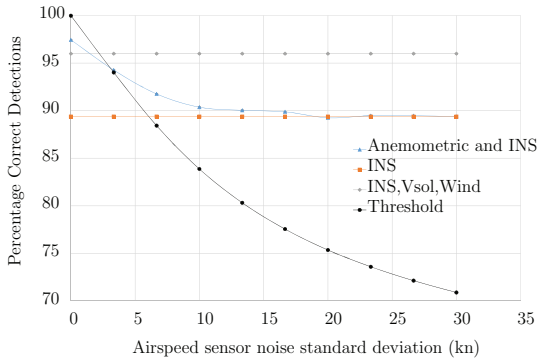

Figure D.19: The SVM's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

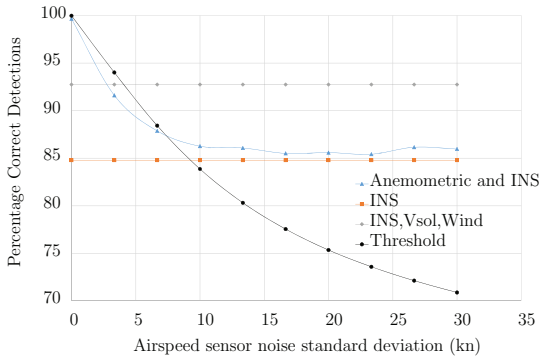

Figure D.20: The DT's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

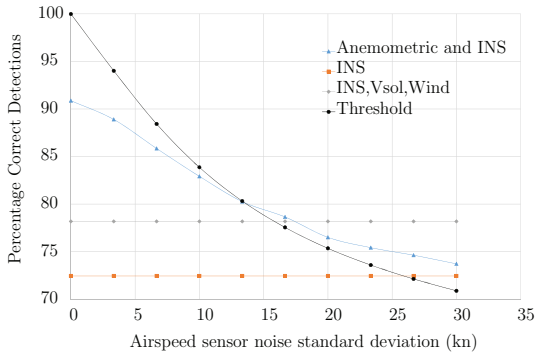

Figure D.21: The NB's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

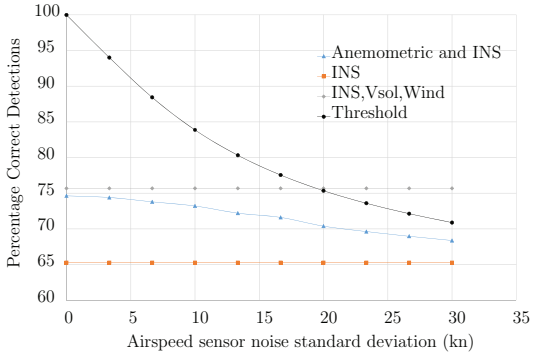

Figure D.22: The knn's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

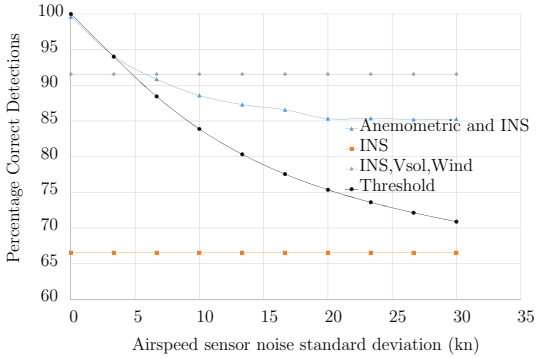

Figure D.23: The AdaBoost's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

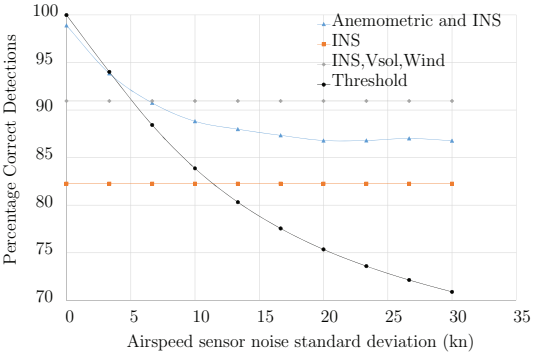

Figure D.24: The RF's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

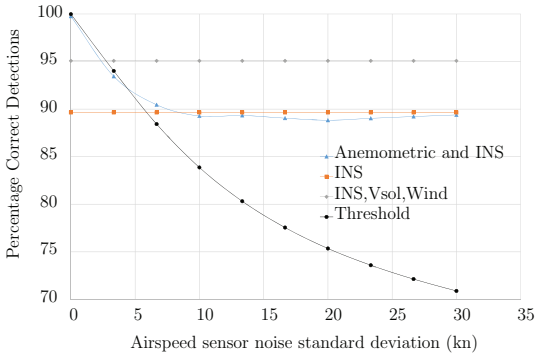

Figure D.25: The bagging's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

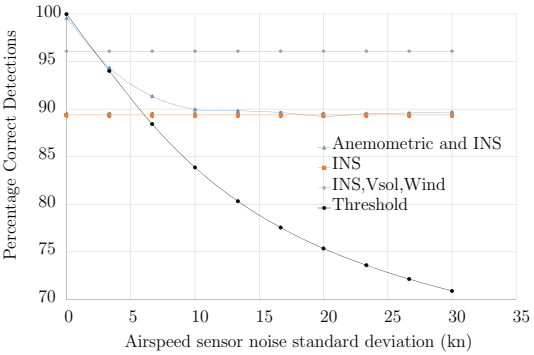

Figure D.26: The MCS's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

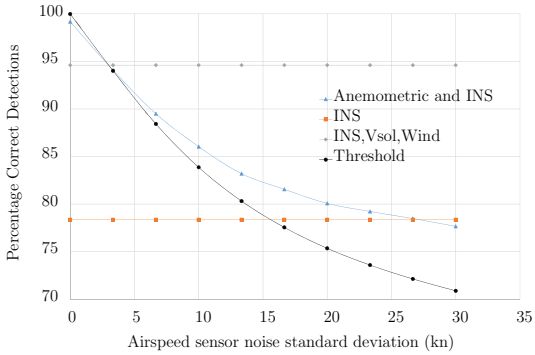

Figure D.27: The LR's accuracy at different noise levels on the airspeed sensor for the three sensor cases.

D.4 False Alarm Distribution

This section shows the airspeed distribution of the false alarms generated by classifiers as well as the airspeed distribution of an Airbus A330 during normal flight at $17500ft$ and $240kn$ under moderate turbulence.

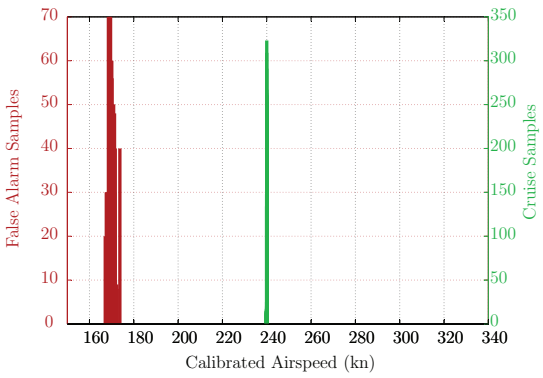

Figure D.28: SVM's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

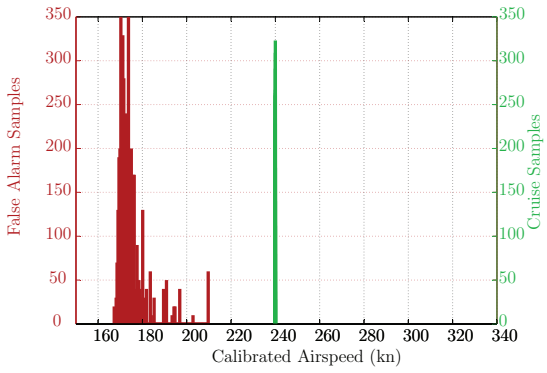

Figure D.29: SVM's false alarm and cruise angle of attack distribution for only the inertial sensors available.

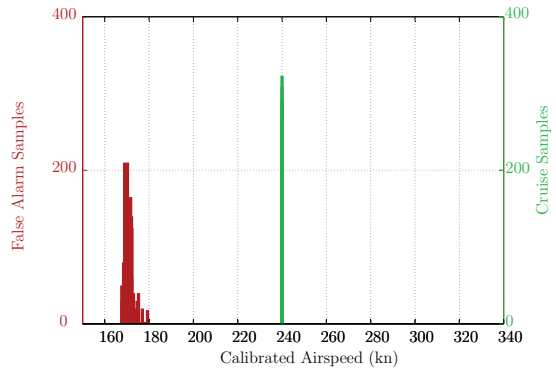

Figure D.30: SVM's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

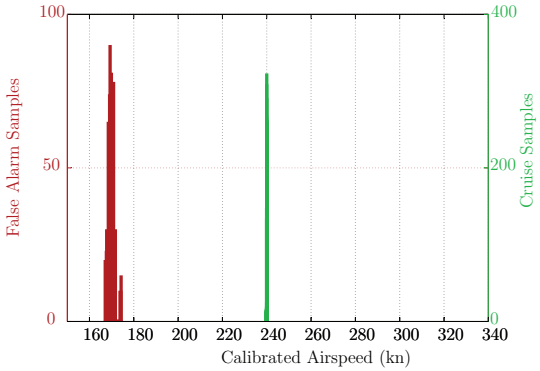

Figure D.31: DT's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

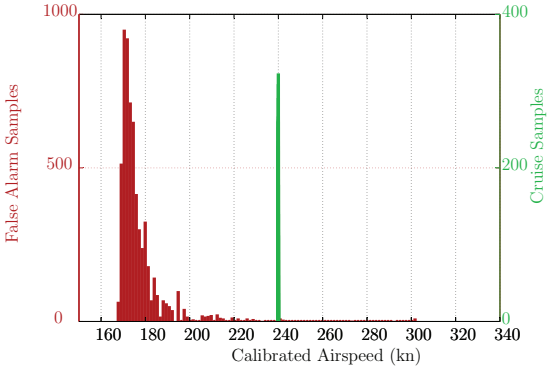

Figure D.32: DT's false alarm and cruise angle of attack distribution for only the inertial sensors available.

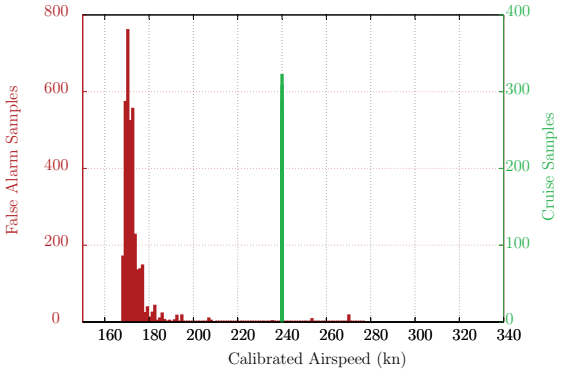

Figure D.33: DT's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

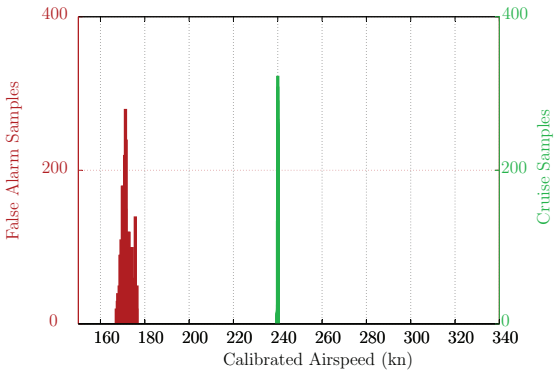

Figure D.34: NB's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

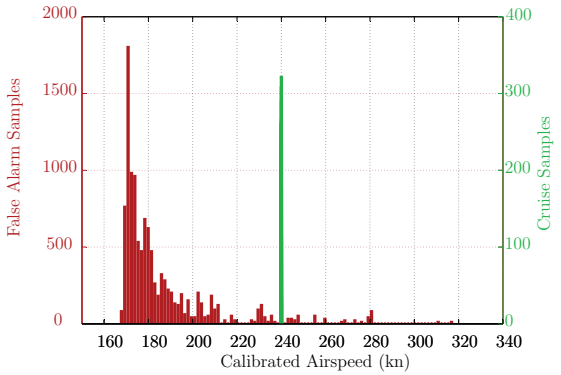

Figure D.35: NB's false alarm and cruise angle of attack distribution for only the inertial sensors available.

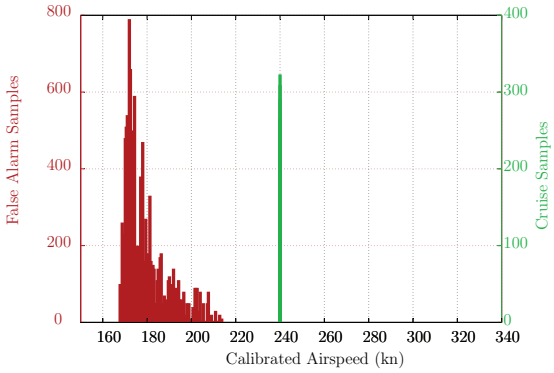

Figure D.36: NB's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

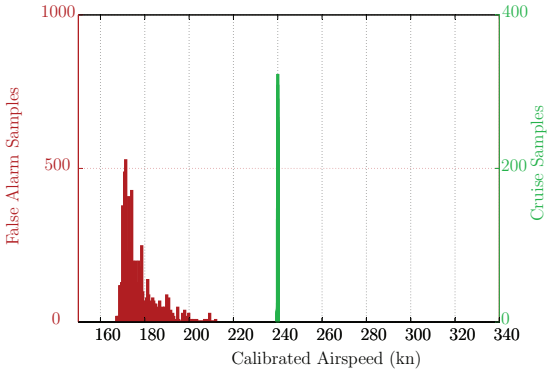

Figure D.37: knn's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

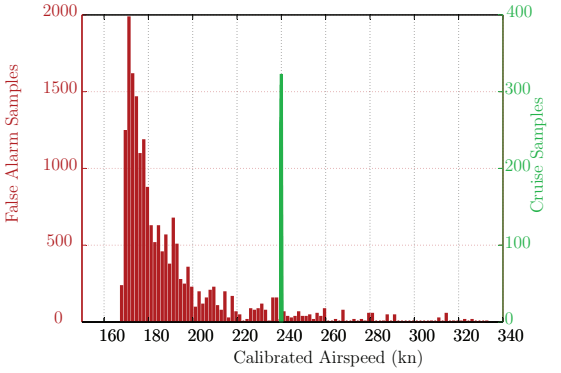

Figure D.38: knn's false alarm and cruise angle of attack distribution for only the inertial sensors available.

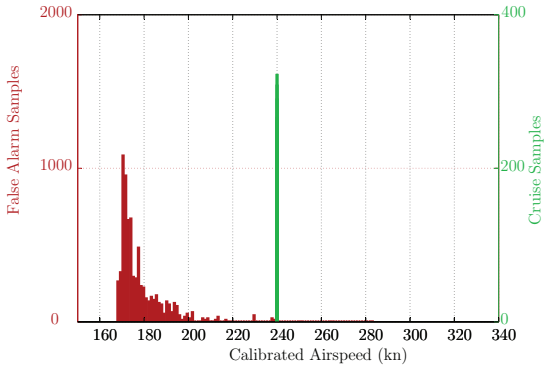

Figure D.39: knn's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

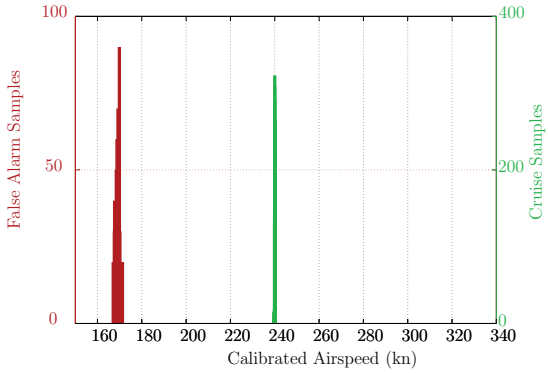

Figure D.40: AdaBoost's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

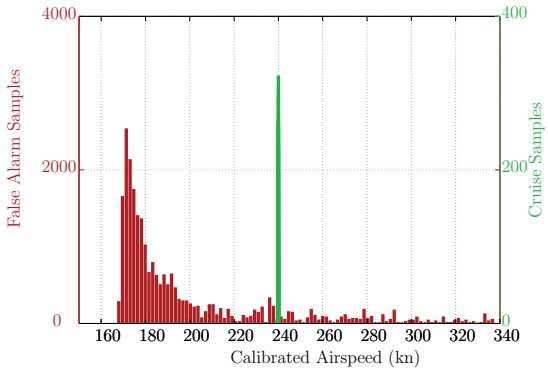

Figure D.41: AdaBoost's false alarm and cruise angle of attack distribution for only the inertial sensors available.

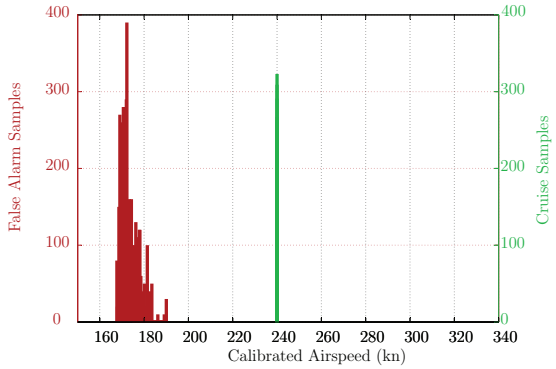

Figure D.42: AdaBoost's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

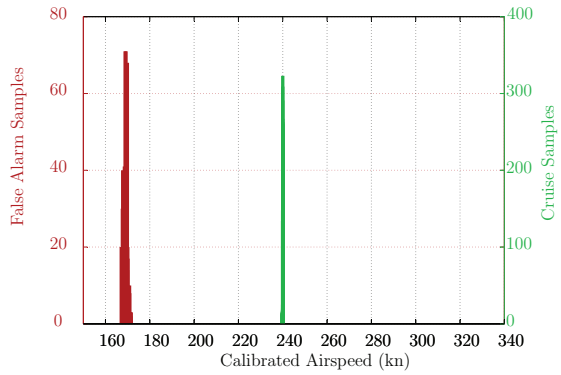

Figure D.43: RF's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

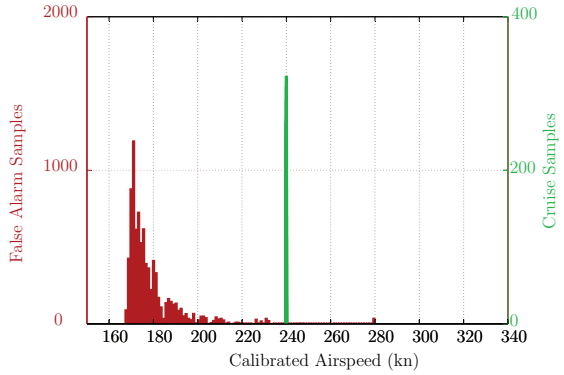

Figure D.44: RF's false alarm and cruise angle of attack distribution for only the inertial sensors available.

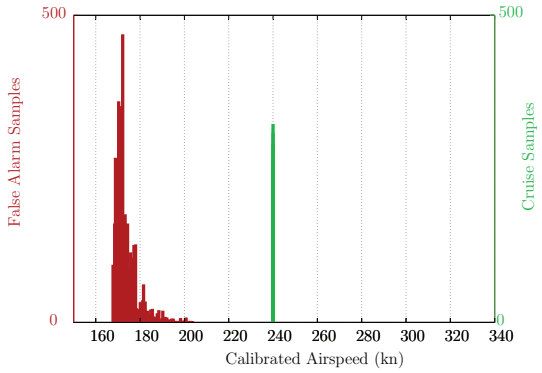

Figure D.45: RF's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

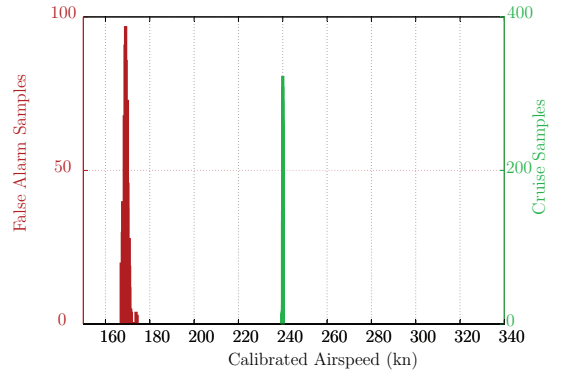

Figure D.46: Bagging's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

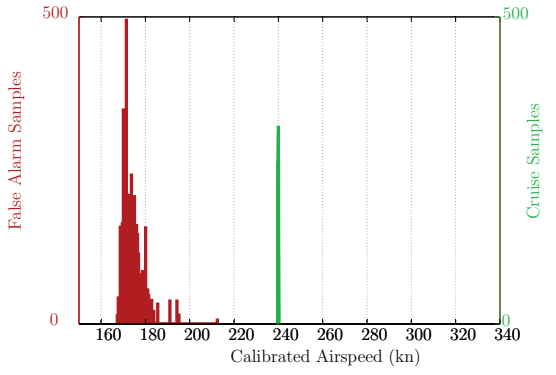

Figure D.47: Bagging's false alarm and cruise angle of attack distribution for only the inertial sensors available.

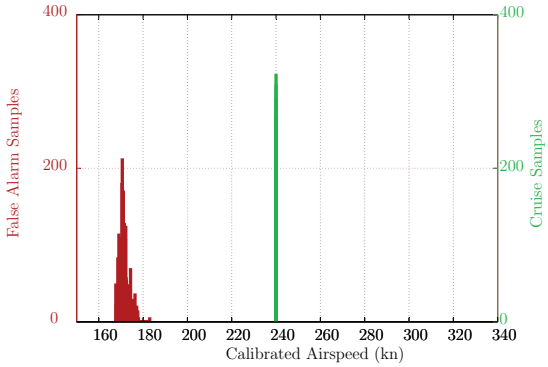

Figure D.48: Bagging's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

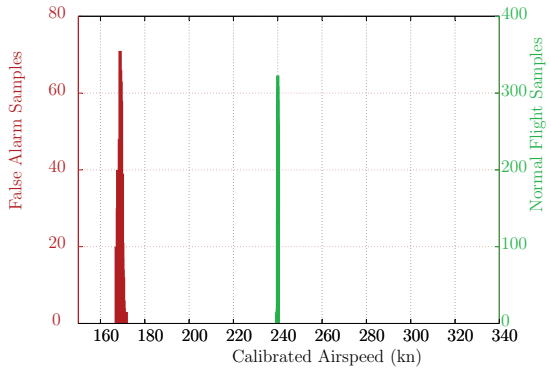

Figure D.49: MCS's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

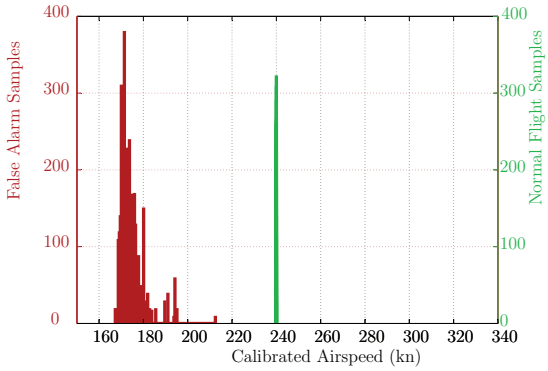

Figure D.50: MCS's false alarm and cruise angle of attack distribution for only the inertial sensors available.

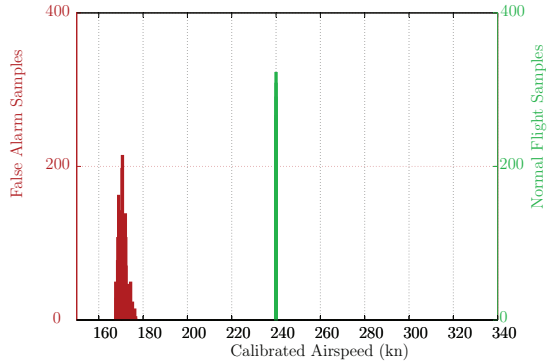

Figure D.51: MCS's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

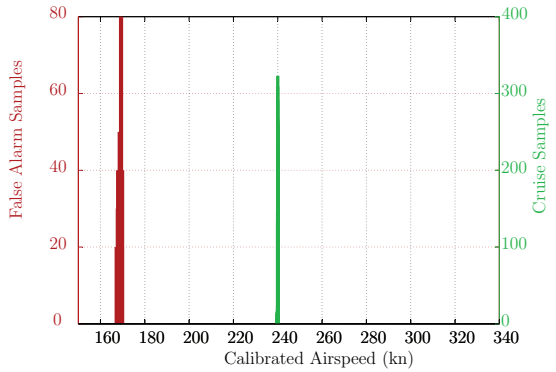

Figure D.52: LR's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

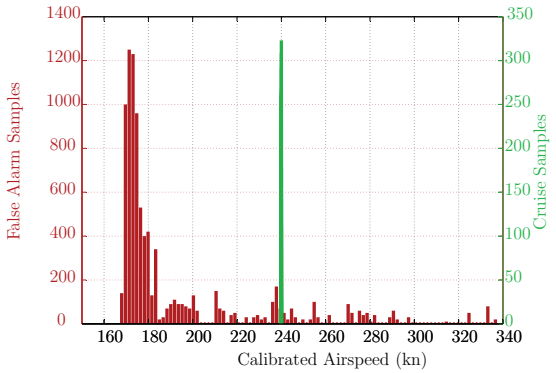

Figure D.53: LR's false alarm and cruise angle of attack distribution for only the inertial sensors available.

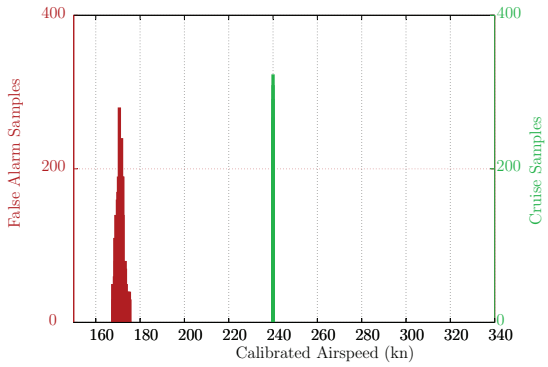

Figure D.54: LR's false alarm and cruise angle of attack distribution for the inertial sensors with ground speed and wind speed available.

D.5 Validation Manoeuvre

This section shows the false alarm and missed detections generated by the classifier during an upset validation manoeuvre. The manoeuvre performed tested three aspects of the classification based upset detection function. Classifiers were tested during manoeuvres that fell within the normal flight operational envelope. The classifiers were secondly tested on manoeuvres that resulted in the aircraft entering the upset domain. Manoeuvres recovering from an upset event were also tested. The position of the false alarms and missed detections from the upset boundary were evaluated. The manoeuvres were used to identify whether the sensor measurements in the training dataset were representative of typical manoeuvres within the upset and normal cruise domains.

The blue line in the figures shows the airspeed trajectory during the manoeuvre. The black line show the upset boundary at the given altitude. False alarms are indicated with red dots and the missed detections are shown with green dots. The majority of the classifiers' false alarms and missed detections when the anemometric and INS sensors were in close vicinity of the upset boundary.

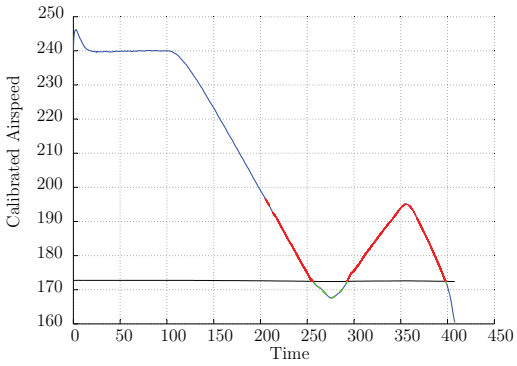

Figure D.55: False alarms and missed detections generated by SVM trained with anemometric and inertial sensor data for an underspeed manoeuvre.

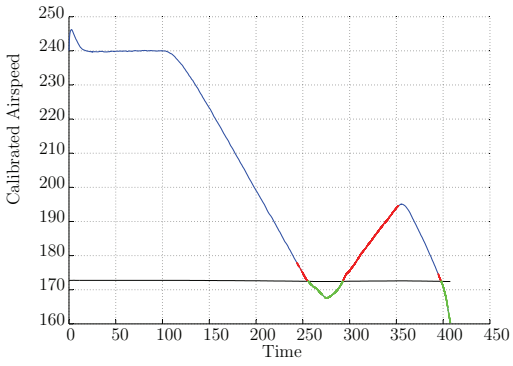

Figure D.56: False alarms and missed detections generated by SVM trained with only inertial sensor data for an underspeed manoeuvre.

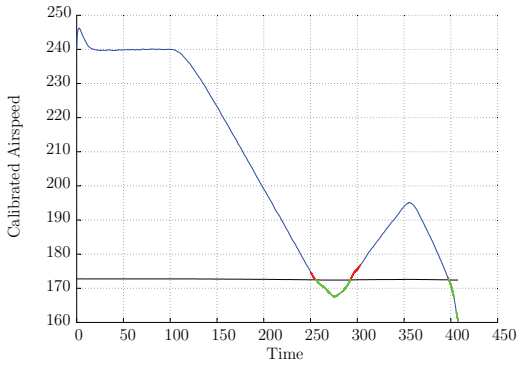

Figure D.57: False alarms and missed detections generated by DT trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

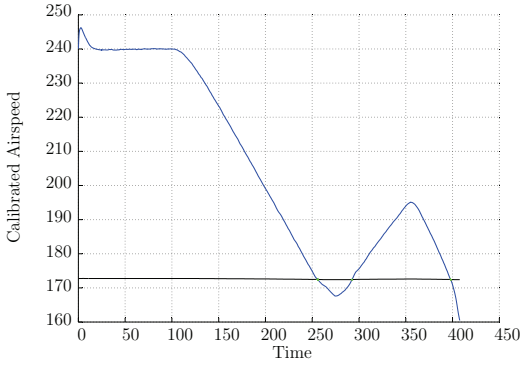

Figure D.58: False alarms and missed detections generated by DT trained with anemometric and inertial sensor data for an underspeed manoeuvre.

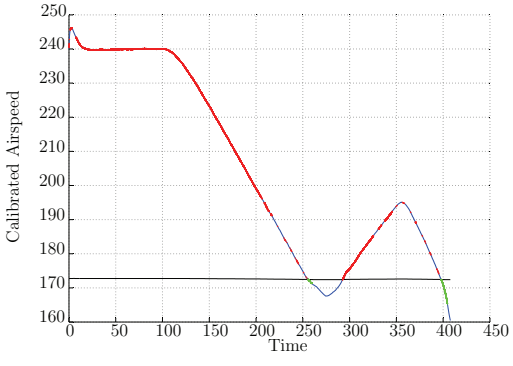

Figure D.59: False alarms and missed detections generated by DT trained with only inertial sensor data for an underspeed manoeuvre.

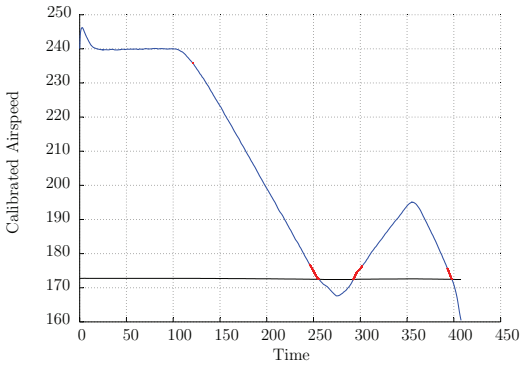

Figure D.60: False alarms and missed detections generated by DT trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

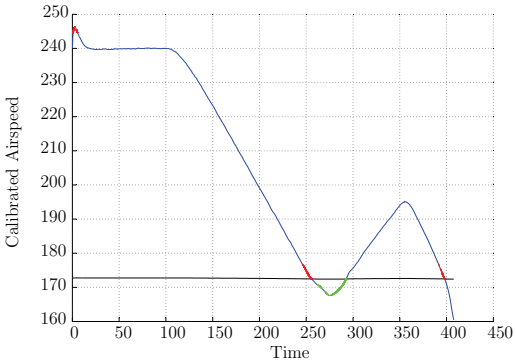

Figure D.61: False alarms and missed detections generated by NB trained with anemometric and inertial sensor data for an underspeed manoeuvre.

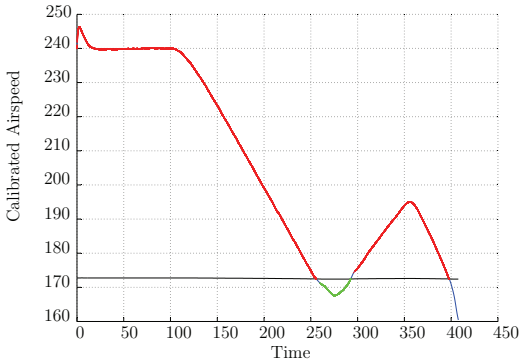

Figure D.62: False alarms and missed detections generated by NB trained with only inertial sensor data for an underspeed manoeuvre.

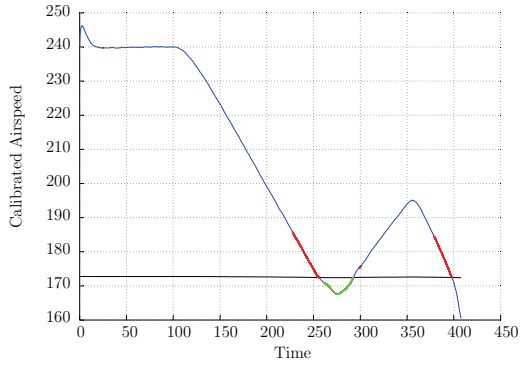

Figure D.63: False alarms and missed detections generated by NB trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

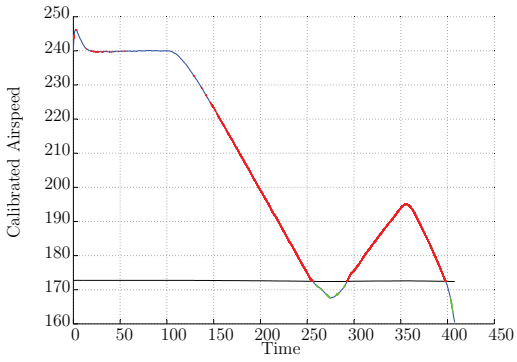

Figure D.64: False alarms and missed detections generated by knn trained with anemometric and inertial sensor data for an underspeed manoeuvre.

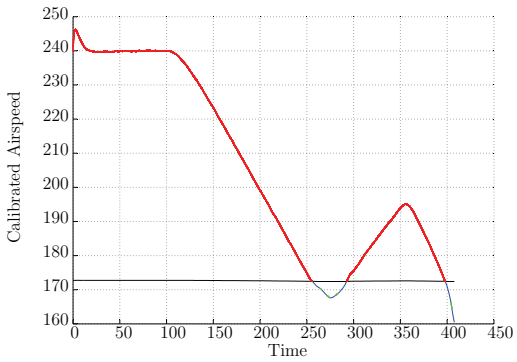

Figure D.65: False alarms and missed detections generated by knn trained with only inertial sensor data for an underspeed manoeuvre.

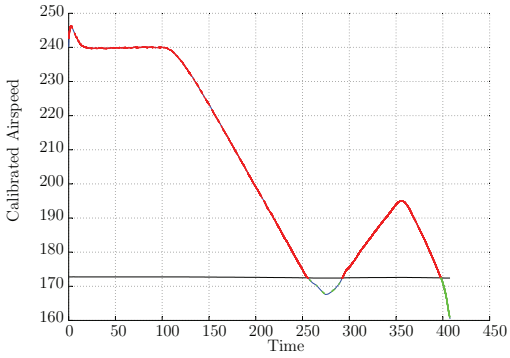

Figure D.66: False alarms and missed detections generated by knn trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

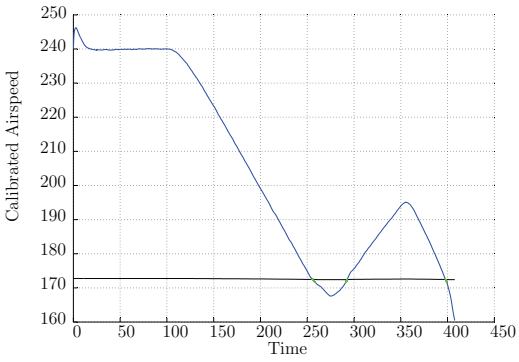

Figure D.67: False alarms and missed detections generated by AdaBoost trained with anemometric and inertial sensor data for an underspeed manoeuvre.

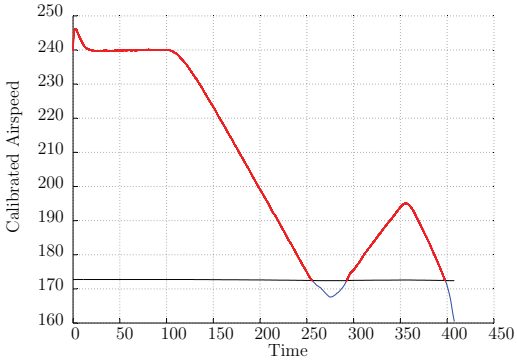

Figure D.68: False alarms and missed detections generated by AdaBoost trained with only inertial sensor data for an underspeed manoeuvre.

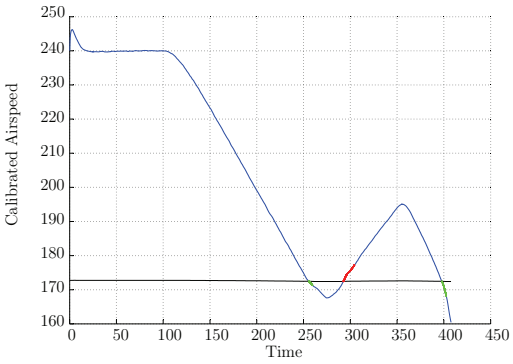

Figure D.69: False alarms and missed detections generated by AdaBoost trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

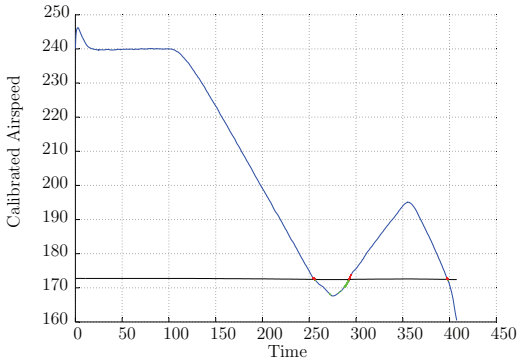

Figure D.70: False alarms and missed detections generated by RF trained with anemometric and inertial sensor data for an underspeed manoeuvre.

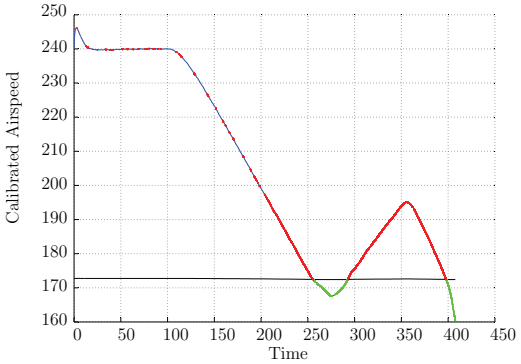

Figure D.71: False alarms and missed detections generated by RF trained with only inertial sensor data for an underspeed manoeuvre.

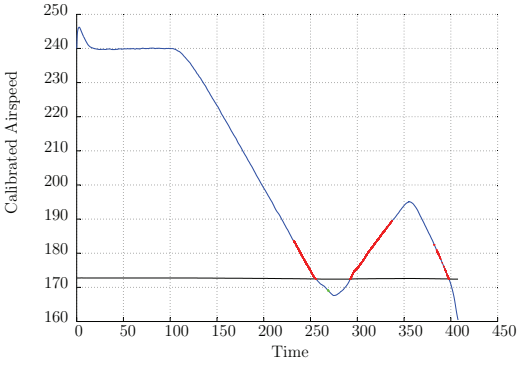

Figure D.72: False alarms and missed detections generated by RF trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

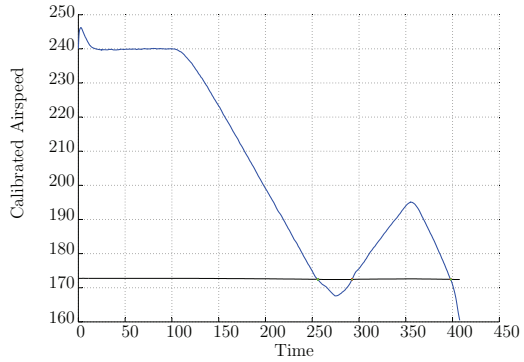

Figure D.73: False alarms and missed detections generated by Bagging trained with anemometric and inertial sensor data for an underspeed manoeuvre.

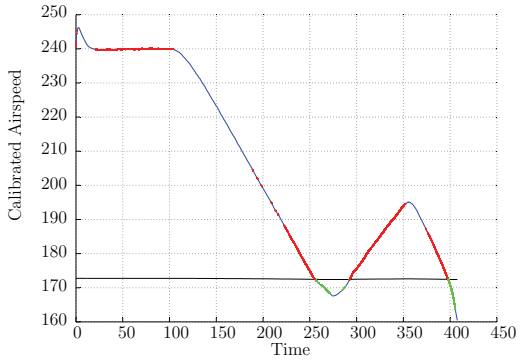

Figure D.74: False alarms and missed detections generated by Bagging trained with only inertial sensor data for an underspeed manoeuvre.

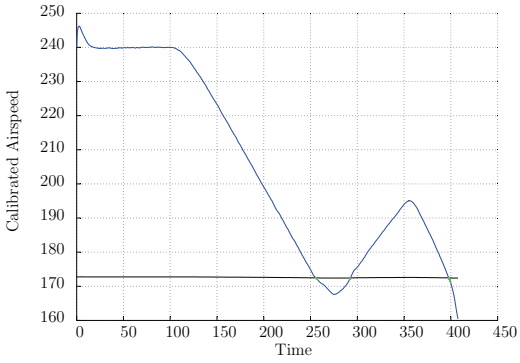

Figure D.75: False alarms and missed detections generated by Bagging trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

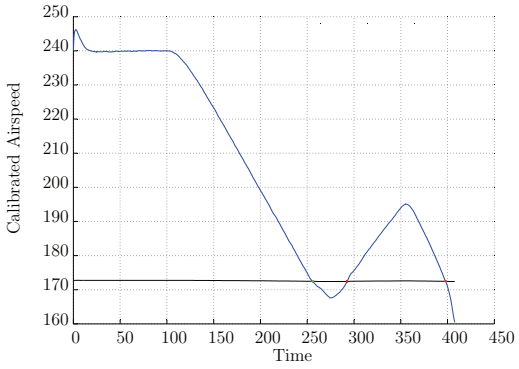

Figure D.76: False alarms and missed detections generated by MCS trained with anemometric and inertial sensor data for an underspeed manoeuvre.

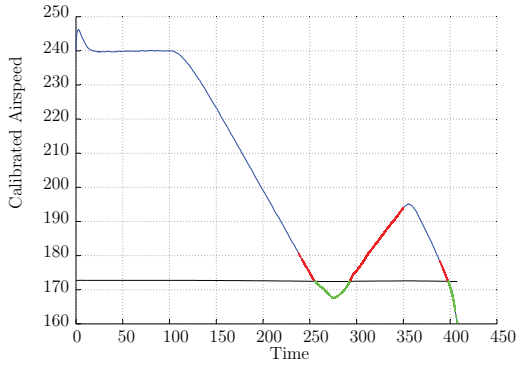

Figure D.77: False alarms and missed detections generated by MCS trained with only inertial sensor data for an underspeed manoeuvre.

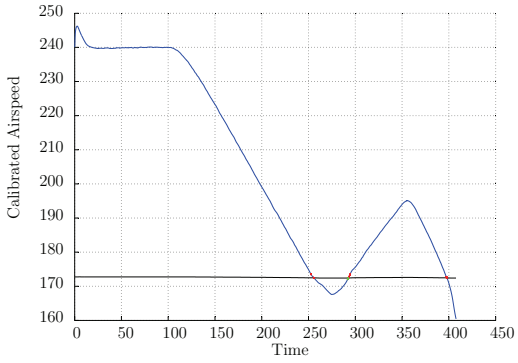

Figure D.78: False alarms and missed detections generated by MCS trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

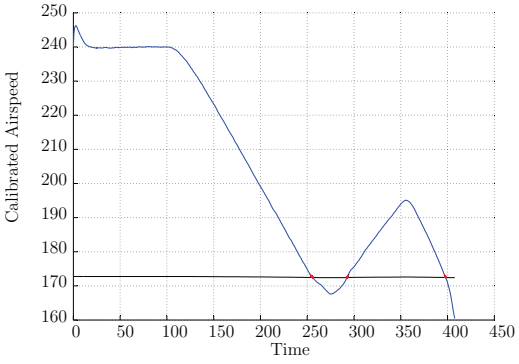

Figure D.79: False alarms and missed detections generated by LR trained with anemometric and inertial sensor data for an underspeed manoeuvre.

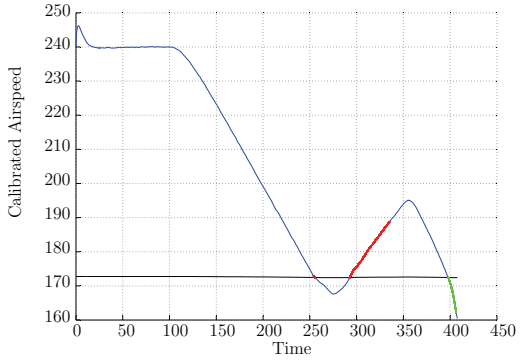

Figure D.80: False alarms and missed detections generated by LR trained with only inertial sensor data for an underspeed manoeuvre.

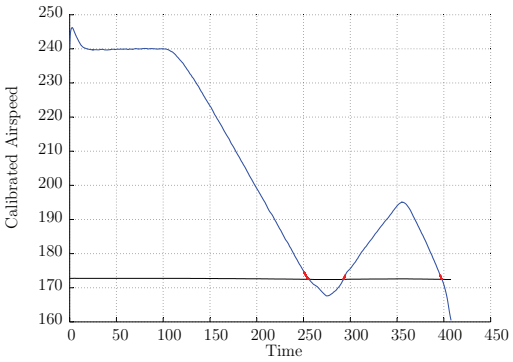

Figure D.81: False alarms and missed detections generated by LR trained with inertial sensor data and estimates of ground and wind speeds for an underspeed manoeuvre.

Appendix E

Additional Dynamic Pitch Upset Classification Results

E.1 Dynamic Pitch Upset Detection Classifier Learning Curve

Learning curves show the classifier accuracy with increasing number of training samples. This was used to ensure that the training dataset was large enough for the classifiers to train properly for dynamic pith upset.

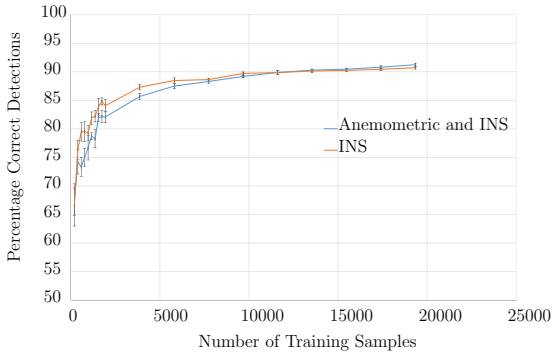

Figure E.1: The SVM's learning curve for both sensor cases.

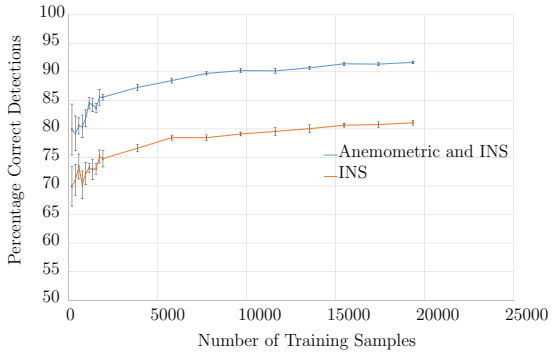

Figure E.2: The DT's learning curve for both sensor cases.

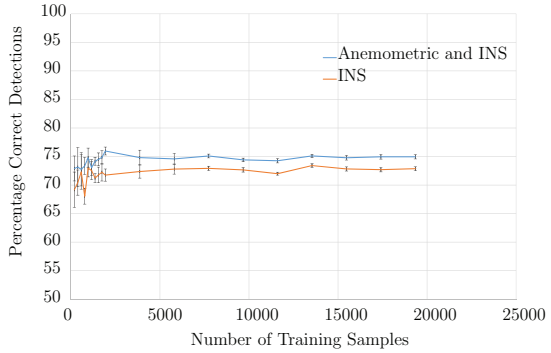

Figure E.3: The NB's learning curve for both sensor cases.

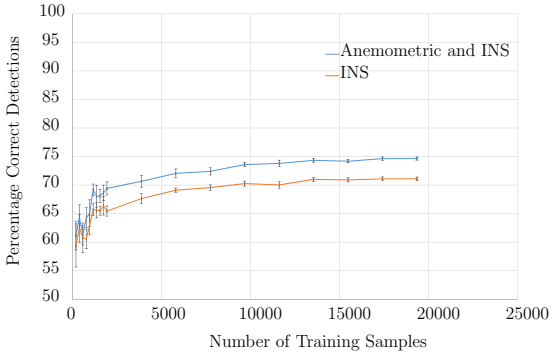

Figure E.4: The knn's learning curve for both sensor cases.

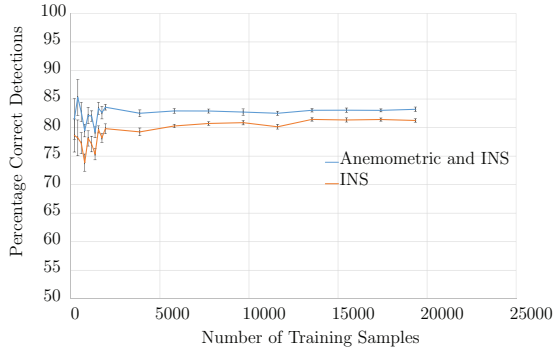

Figure E.5: The AdaBoost's learning curve for both sensor cases.

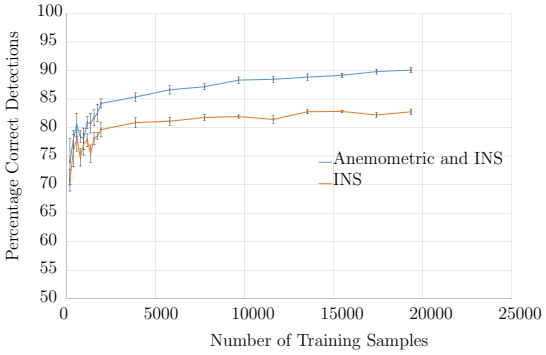

Figure E.6: The RF's learning curve for both sensor cases.

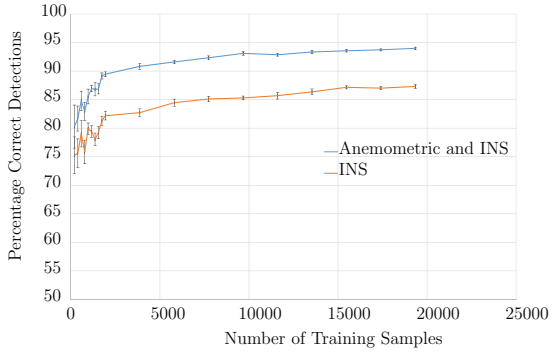

Figure E.7: The Bagging's learning curve for both sensor cases.

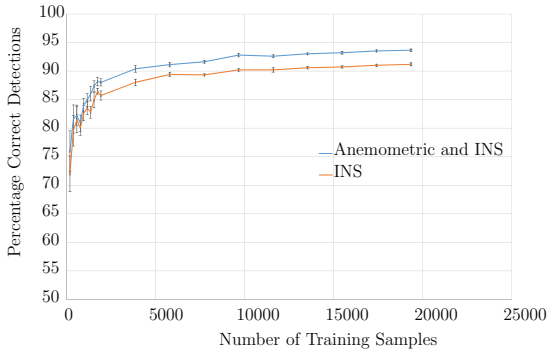

Figure E.8: The MCS's learning curve for both sensor cases.

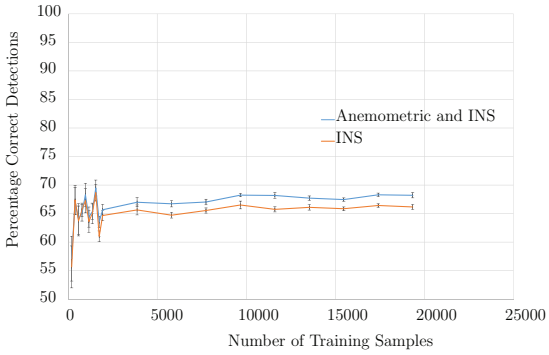

Figure E.9: The LR's learning curve for both sensor cases.

E.2 Receiver Operator Characterisation Curves for Classifiers

The receiver operating characteristic curves are shown for all the classifiers trained to detect dynamic pitch upset. The ROC curves for both sensor cases are shown on the same figure. These curves offer insight into the false alarms and missed detections generated at different thresholds on the probability outputs of the classifiers. The classifiers are required to perform at an equal error rate, the ROC curves are used to determine the desired threshold to ensure an equal error rate. It is clear from the ROC curves shown that the thresholds were typically near a probability of 0.5. The ROC curves can be used to bias the classifiers towards missed detections to reduce the amount of false alarms generated.

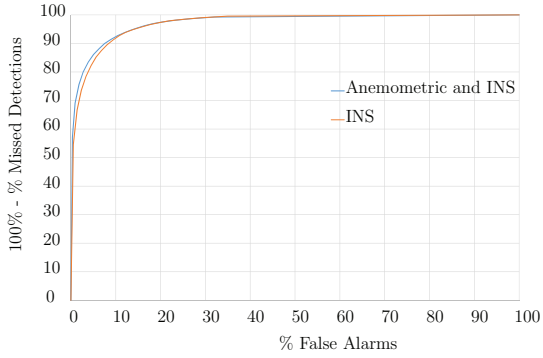

Figure E.10: The SVM's ROC curve for both sensor cases.

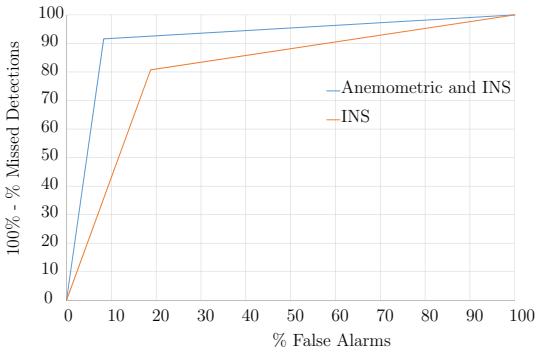

Figure E.11: The DT's ROC curve for both sensor cases.

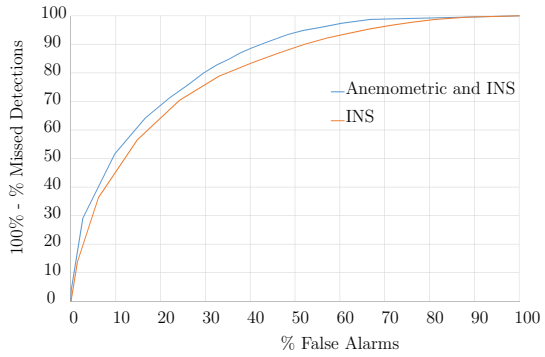

Figure E.12: The NB's ROC curve for both sensor cases.

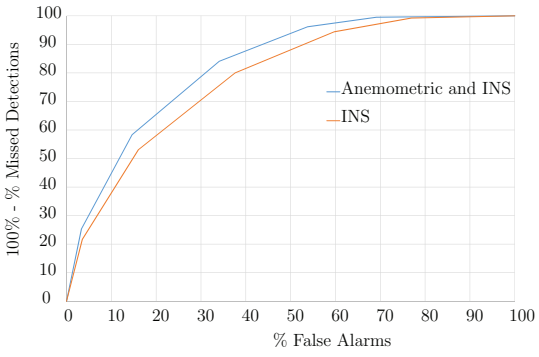

Figure E.13: The knn's ROC curve for both sensor cases.

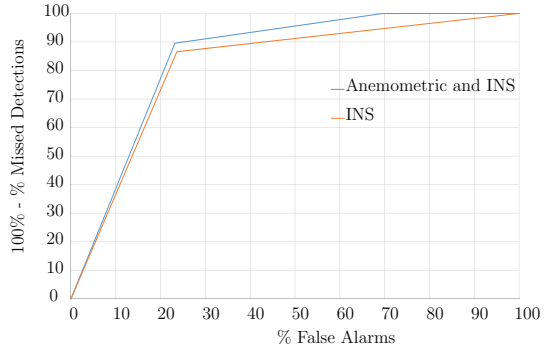

Figure E.14: The AdaBoost's ROC curve for both sensor cases.

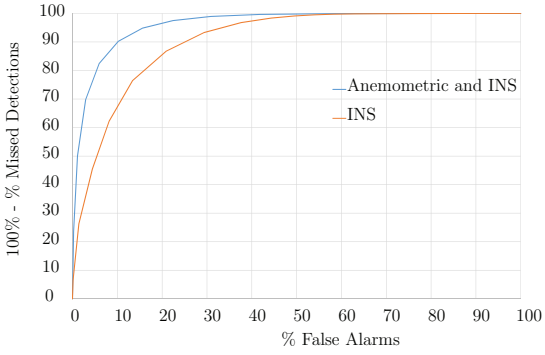

Figure E.15: The RF's ROC curve for both sensor cases.

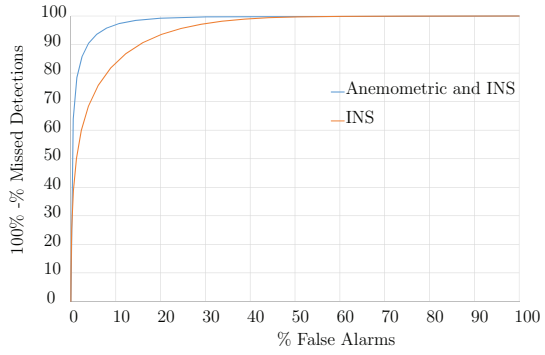

Figure E.16: The Bagging's ROC curve for both sensor cases.

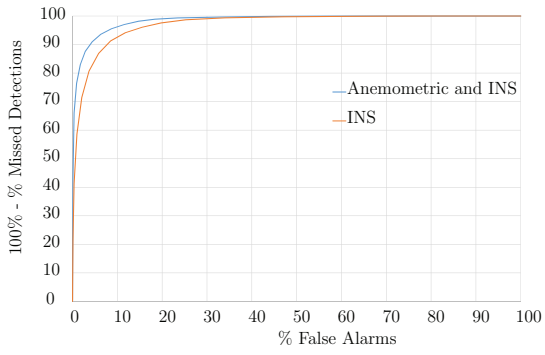

Figure E.17: The MCS's ROC curve for both sensor cases.

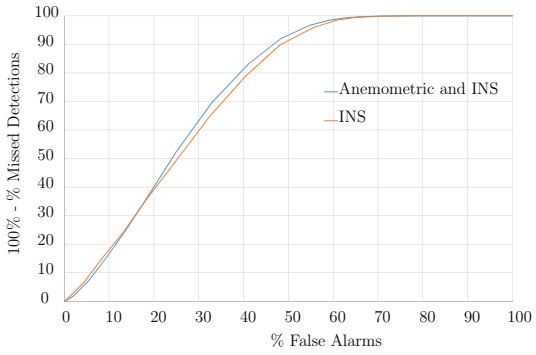

Figure E.18: The LR's ROC curve for both sensor cases.

E.3 Classifier Noise Accuracy

The classifier accuracies at different noise levels on the angle of attack sensor are shown in Figure E.19 to E.27. The noise on the angle of attack was assumed to be a zero mean gaussian distribution. The standard deviation of the noise were varied between 0 degrees to 5 degrees and the percentage correct detections were calculated.

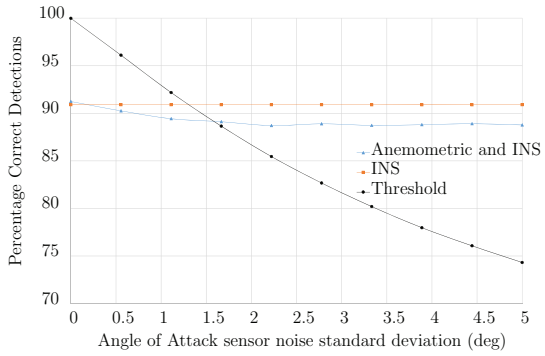

Figure E.19: The SVM's accuracy at different sensor noise levels for both sensor cases.

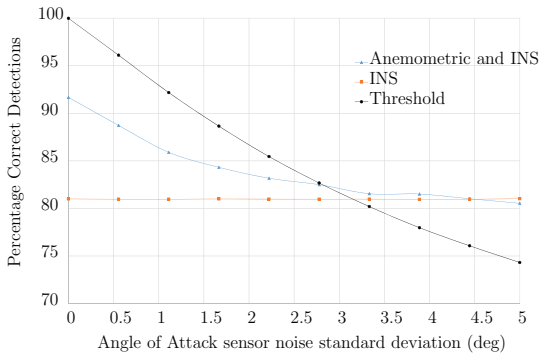

Figure E.20: The DT's accuracy at different sensor noise levels for both sensor cases.

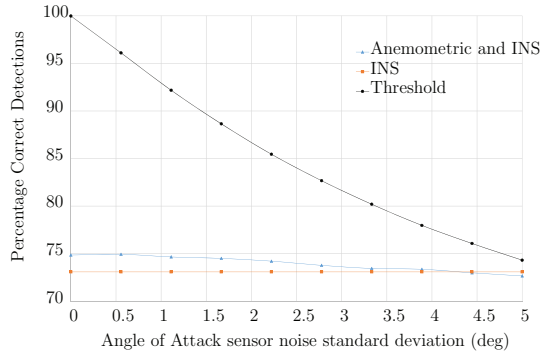

Figure E.21: The NB's accuracy at different sensor noise levels for both sensor cases.

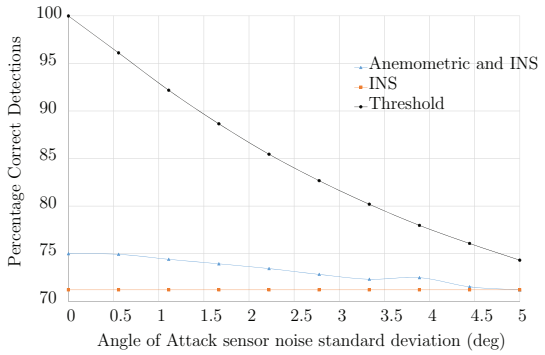

Figure E.22: The knn's accuracy at different sensor noise levels for both sensor cases.

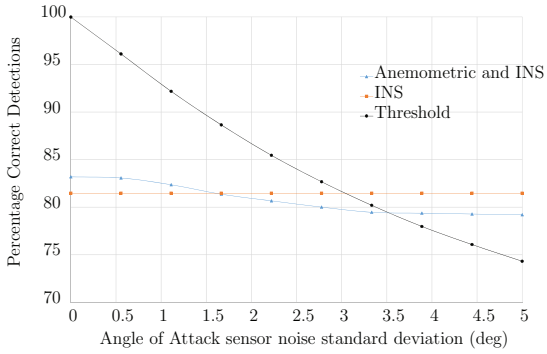

Figure E.23: The AdaBoost's accuracy at different sensor noise levels for both sensor cases.

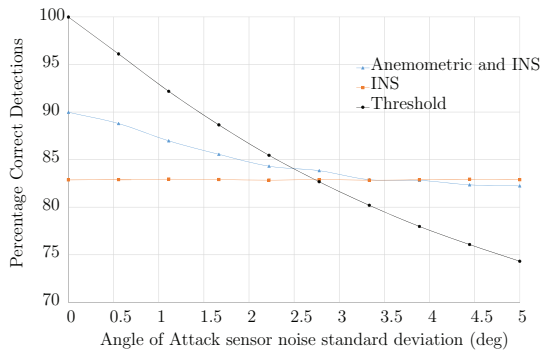

Figure E.24: The RF's accuracy at different sensor noise levels for both sensor cases.

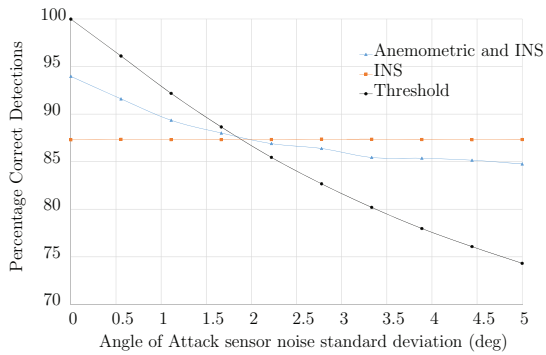

Figure E.25: The Bagging's accuracy at different sensor noise levels for both sensor cases.

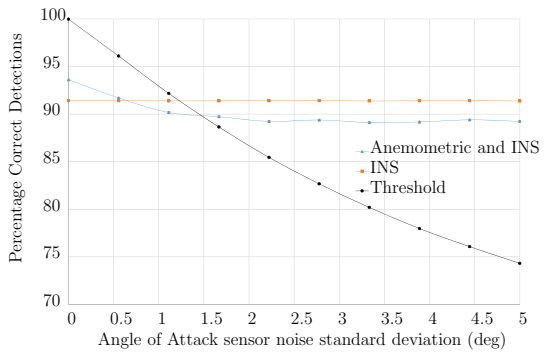

Figure E.26: The MCS's accuracy at different sensor noise levels for both sensor cases.

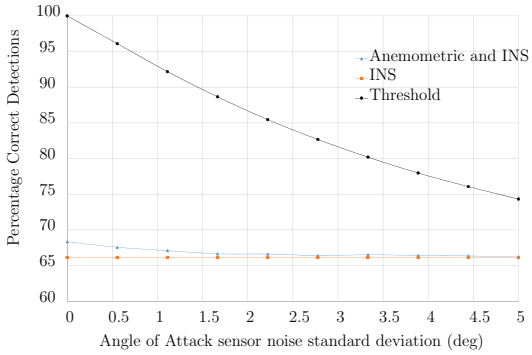

Figure E.27: The LR's accuracy at different sensor noise levels for both sensor cases.

E.4 False Alarm Position

This section shows the angle of attack and pitch rate distributions of the false alarms generated by classifiers as well as the angle of attack and pitch rate distributions of an Airbus A330 during normal flight at $17500ft$ and $240kn$ under moderate turbulence.

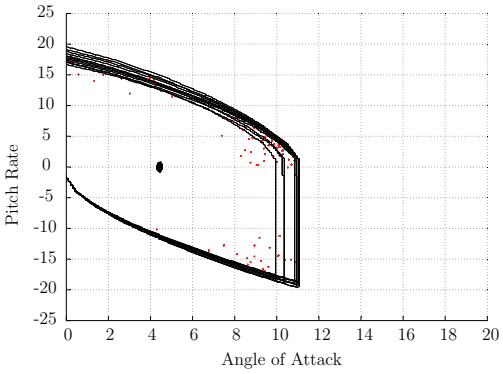

Figure E.28: SVM's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

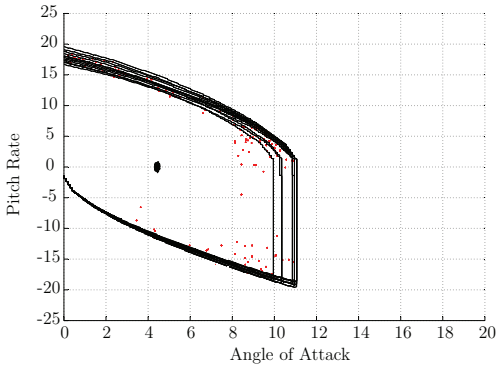

Figure E.29: SVM's false alarm and cruise angle of attack distribution for only the inertial sensors available.

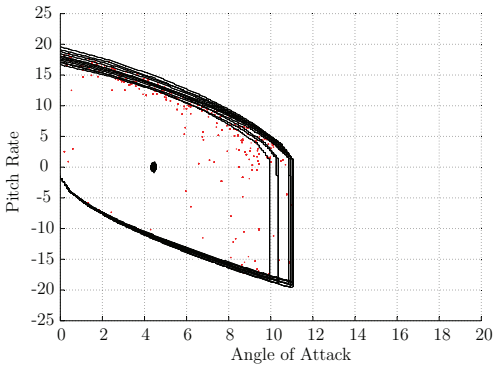

Figure E.30: DT's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

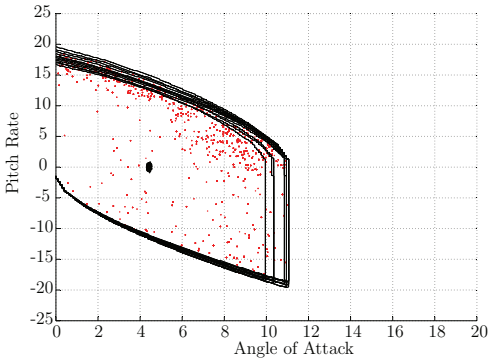

Figure E.31: DT's false alarm and cruise angle of attack distribution for only the inertial sensors available.

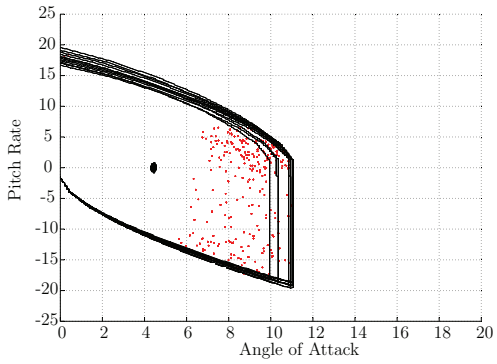

Figure E.32: NB's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

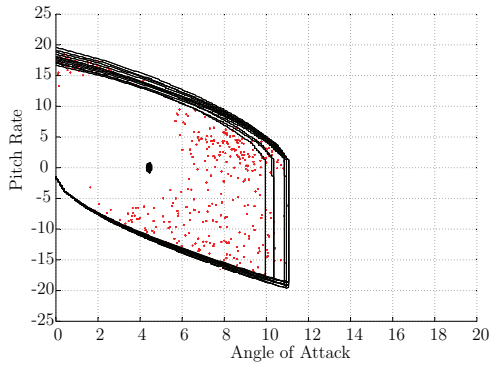

Figure E.33: NB's false alarm and cruise angle of attack distribution for only the inertial sensors available.

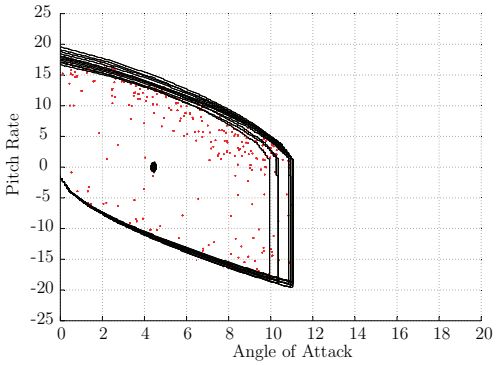

Figure E.34: knn's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

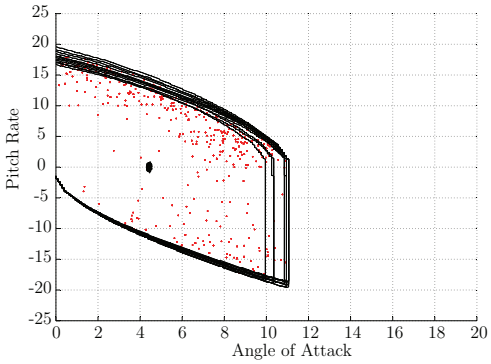

Figure E.35: knn's false alarm and cruise angle of attack distribution for only the inertial sensors available.

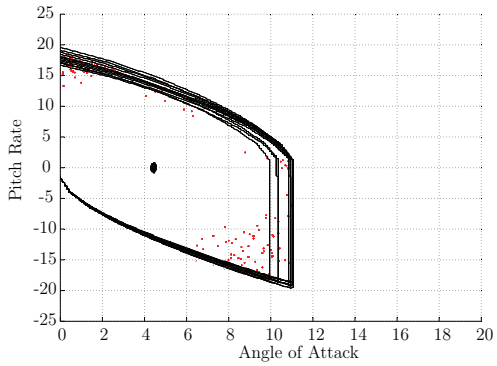

Figure E.36: AdaBoost's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

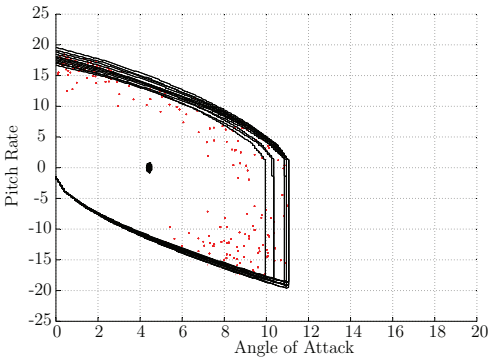

Figure E.37: AdaBoost's false alarm and cruise angle of attack distribution for only the inertial sensors available.

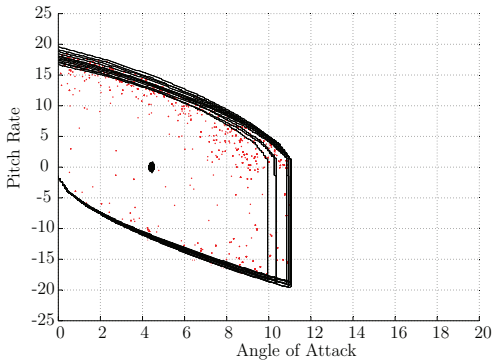

Figure E.38: RF's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

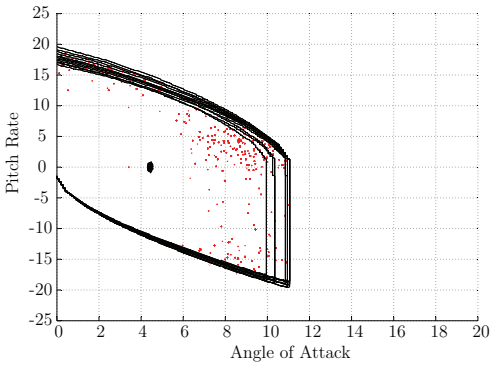

Figure E.39: RF's false alarm and cruise angle of attack distribution for only the inertial sensors available.

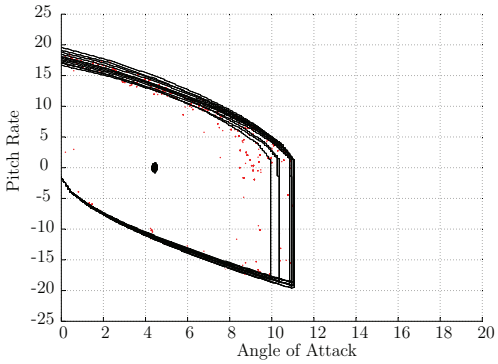

Figure E.40: Bagging's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

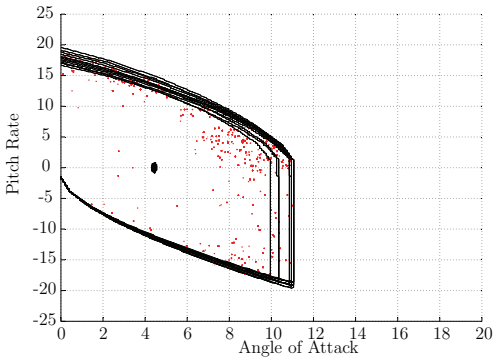

Figure E.41: Bagging's false alarm and cruise angle of attack distribution for only the inertial sensors available.

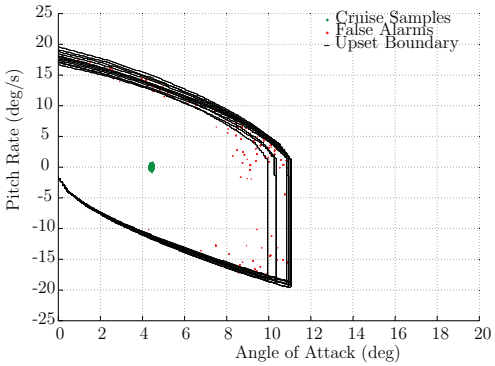

Figure E.42: MCS's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

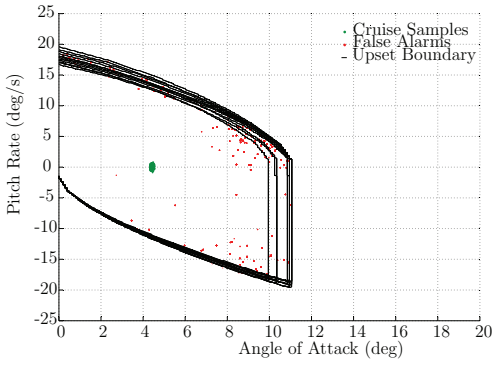

Figure E.43: MCS's false alarm and cruise angle of attack distribution for only the inertial sensors available.

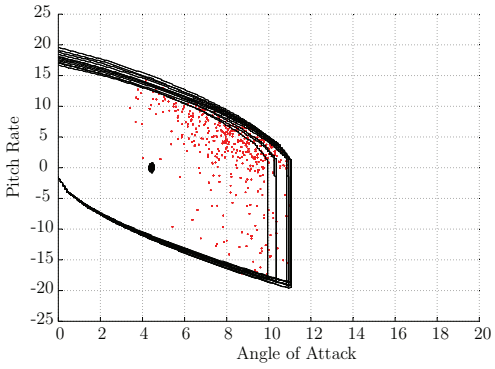

Figure E.44: LR's false alarm and cruise angle of attack distribution for the anemometric and inertial sensors available.

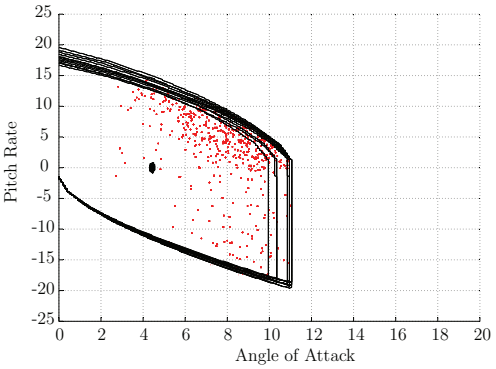

Figure E.45: LR's false alarm and cruise angle of attack distribution for only the inertial sensors available.

E.5 Validation Manoeuvre

This section shows the false alarm and missed detections generated by the classifier during an upset validation manoeuvre. The manoeuvre performed tested three aspects of the classification based upset detection function. Classifiers were tested during manoeuvres that fell within the normal flight operational envelope. The classifiers were secondly tested om manoeuvres that resulted in the aircraft entering the upset domain. Manoeuvres recovering from an upset event were lastly tested. The position of the false alarms and missed detections from the upset boundary were evaluated. The manoeuvres were used to identify whether the sensor measurements in the training dataset were representative of typical manoeuvres within the upset and normal cruise domains.

The blue and red line in the figures shows the angle of attack and pitch rate trajectory during the manoeuvre. The black line show the upset boundary for the given altitude

and airspeeds. The false alarms are indicated with red dots and the missed detections are indicated with green dots.

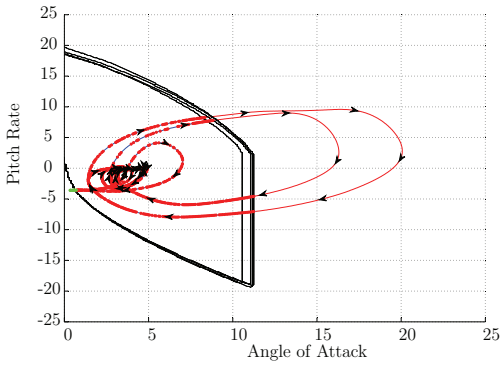

Figure E.46: False alarms and missed detections generated by SVM trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

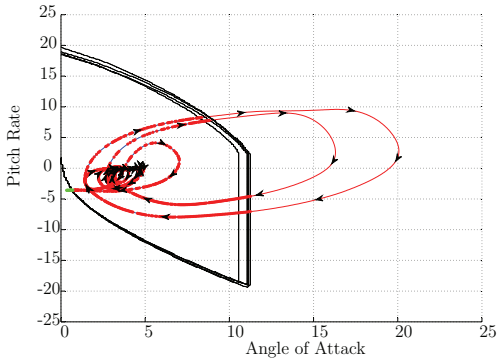

Figure E.47: False alarms and missed detections generated by SVM trained with only inertial sensor data for a dynamic pitch manoeuvre.

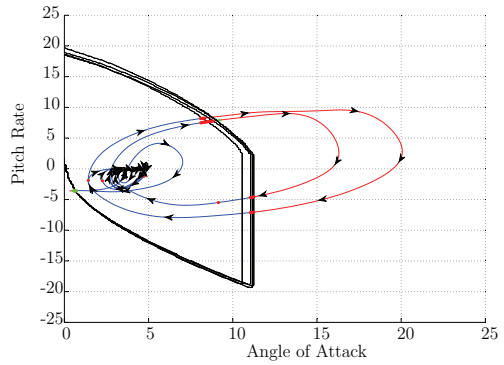

Figure E.48: False alarms and missed detections generated by DT trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

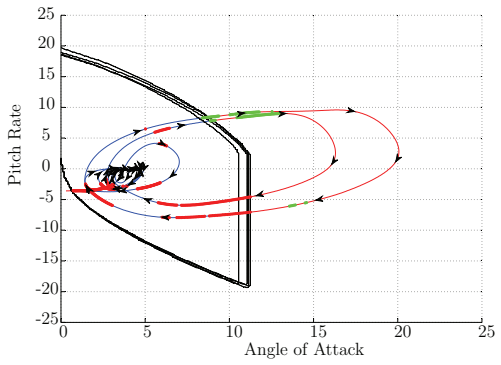

Figure E.49: False alarms and missed detections generated by DT trained with only inertial sensor data for a dynamic pitch manoeuvre.

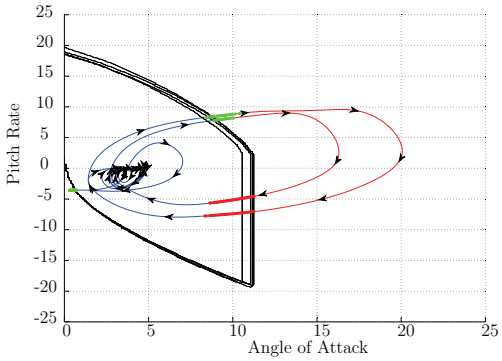

Figure E.50: False alarms and missed detections generated by NB trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

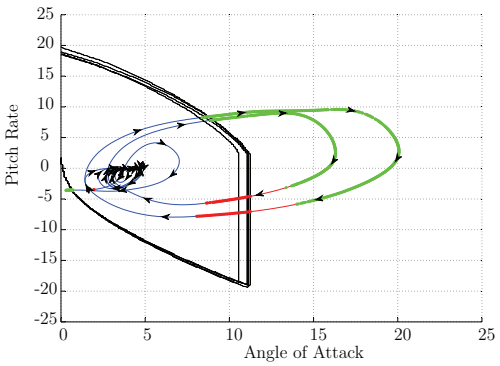

Figure E.51: False alarms and missed detections generated by NB trained with only inertial sensor data for a dynamic pitch manoeuvre.

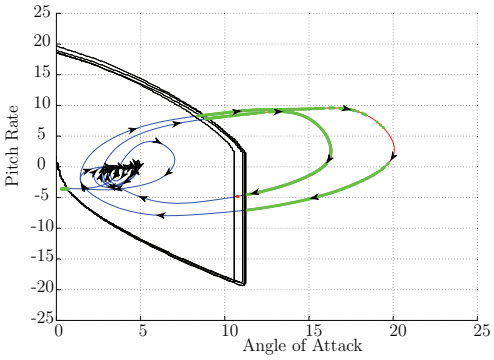

Figure E.52: False alarms and missed detections generated by knn trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

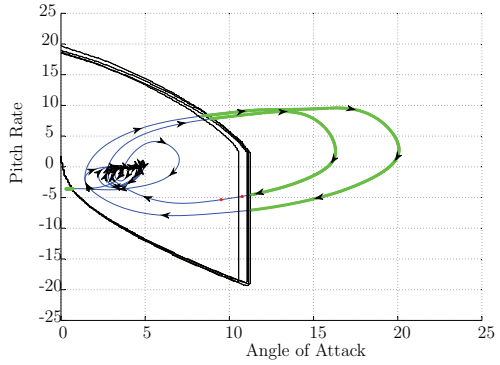

Figure E.53: False alarms and missed detections generated by knn trained with only inertial sensor data for a dynamic pitch manoeuvre.

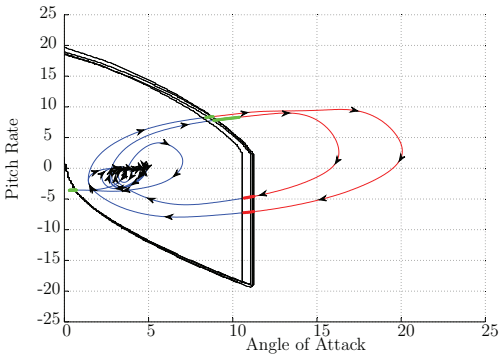

Figure E.54: False alarms and missed detections generated by AdaBoost trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

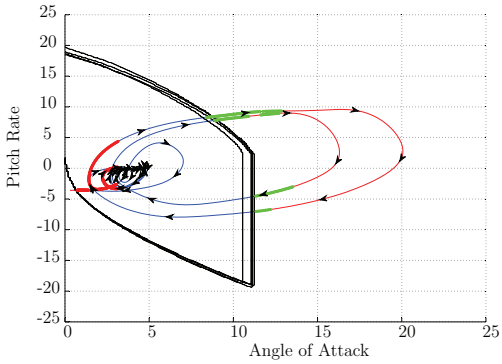

Figure E.55: False alarms and missed detections generated by AdaBoost trained with only inertial sensor data for a dynamic pitch manoeuvre.

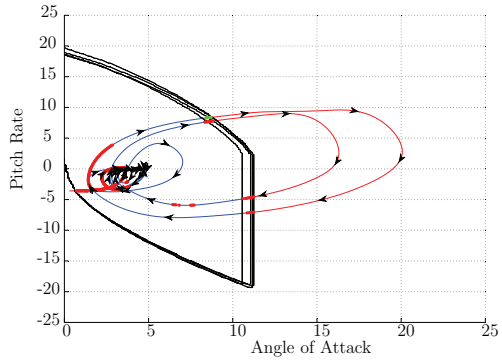

Figure E.56: False alarms and missed detections generated by RF trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

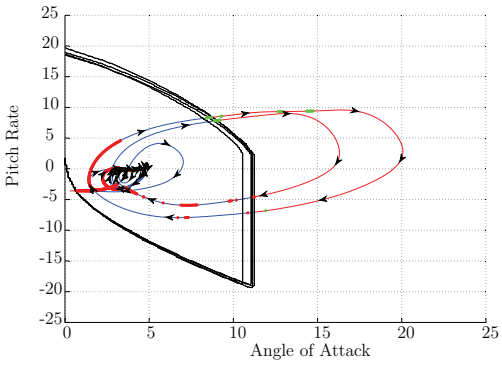

Figure E.57: False alarms and missed detections generated by RF trained with only inertial sensor data for a dynamic pitch manoeuvre.

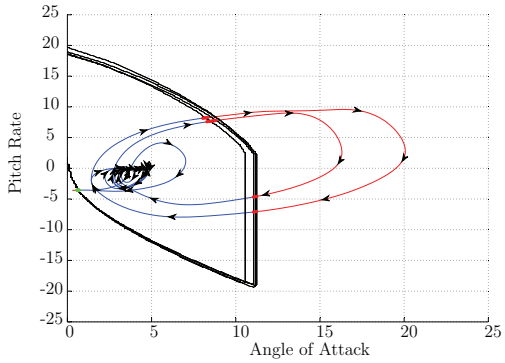

Figure E.58: False alarms and missed detections generated by Bagging trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

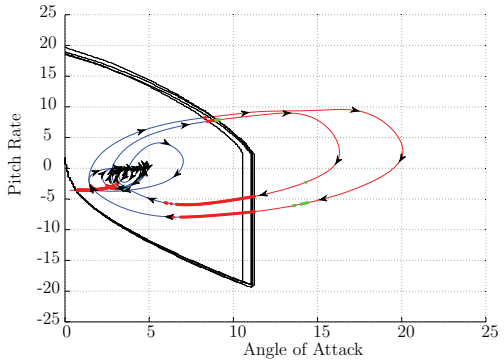

Figure E.59: False alarms and missed detections generated by Bagging trained with only inertial sensor data for a dynamic pitch manoeuvre.

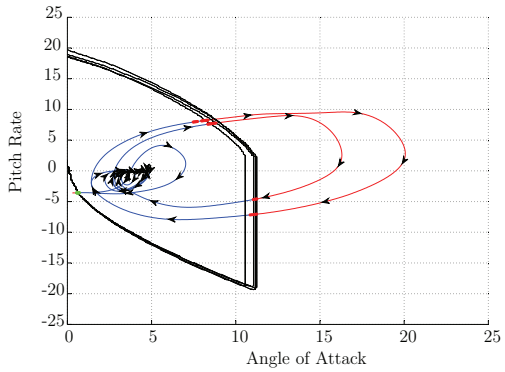

Figure E.60: False alarms and missed detections generated by MCS trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

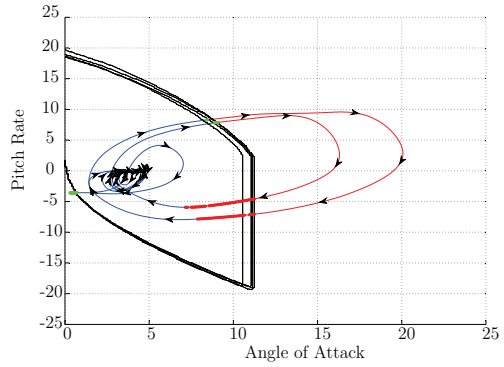

Figure E.61: False alarms and missed detections generated by MCS trained with only inertial sensor data for a dynamic pitch manoeuvre.

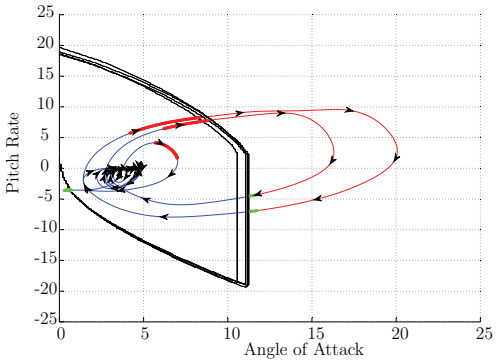

Figure E.62: False alarms and missed detections generated by LR trained with anemometric and inertial sensor data for a dynamic pitch manoeuvre.

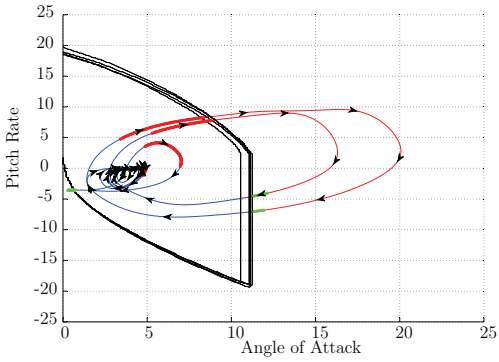

Figure E.63: False alarms and missed detections generated by LR trained with only inertial sensor data for a dynamic pitch manoeuvre.